Altova XMLSpy 2021 Professional Edition

ユーザーマニュアル

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

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

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

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

公開日: 2015-2021

(C) 2015-2021 Altova GmbH

目次

1 XMLSpy とこのドキュメントについて 22

1.1 新規機能 2021 ... 23
1.1.1 バージョン 2020 ... 23
1.1.2 バージョン 2019 ... 24
1.1.3 バージョン 2018 ... 24
1.1.4 バージョン 2017 ... 25
1.2 Windows ファイルパス ... 27
1.3 RaptorXML Server について ... 28

2 XMLSpy チュートリアル 29

2.1 XMLSpy インターフェイス ... 30
2.1.1 表示 ... 31
2.1.2 ウィンドウ ... 32
2.1.3 メニューとツールバー ... 34
2.1.4 テキストビューの設定 ... 36
2.1.5 アプリケーションオプション ... 38
2.2 XML スキーマ：基本 ... 41
2.2.1 新しい XML スキーマファイルの作成 ... 41
2.2.2 名前空間の定義 ... 43
2.2.3 コンテンツモデルの定義 ... 44
2.2.4 ドラッグアンドドロップによる要素の追加 ... 49
2.2.5 コンテンツモデルビューの構成 ... 50
2.2.6 基本的なスキーマの完成 ... 52
2.3 XML スキーマ: 高度な機能 ... 56
2.3.1 複合型と単純型を扱う ... 56
2.3.2 グローバル要素の参照 ... 63
2.3.3 属性と列挙型 ... 65
2.4 XML スキーマ：XMLSpy の機能 ... 68
2.4.1 スキーマナビゲーション ... 68

2.4.2 スキーマドキュメンテーション 70
2.5 XML ドキュメント 74
2.5.1 新規 XML ファイルの作成 74
2.5.2 要素の型の指定 76
2.5.3 グリッドビューを使ったデータの入力 78
2.5.4 テキストビューを使ったデータの入力 79
2.5.5 ドキュメントの検証 83
2.5.6 要素と属性の追加 87
2.5.7 データベース/テーブルビューでの編集 89
2.5.8 スキーマの修正 93
2.6 XSLT 変換 94
2.6.1 XSLT ファイルの割り当て 94
2.6.2 XMLファイルの変換 95
2.6.3 XSLファイルの修正 96
2.7 プロジェクト管理 98
2.7.1 プロジェクトの利点 98
2.7.2 プロジェクトの作成 98
2.8 おしまい 101

3 インターフェイスと環境 102
3.1 グラフィカルユーザインターフェース (GUI) 103
3.1.1 メイン ウィンドウ 104
3.1.2 プロジェクト ウィンドウ 105
3.1.3 情報ウィンドウ 107
3.1.4 入力ヘルパー 107
3.1.5 出力ウィンドウ: XPath/XQuery 108
3.1.6 出力ウィンドウ:XSL アウトライン 109
3.1.7 出力ウィンドウ: HTTP 110
3.1.8 出力ウィンドウ:ファイルの中を検索 110
3.1.9 出力ウィンドウ:スキーマの中を検索 112
3.1.10 メニューバー、ツールバー、ステータスバー 112
3.2 アプリケーション環境 114
3.2.1 設定、カスタマイズ 114
3.2.2 チュートリアル、プロジェクト、サンプル 116

3.2.3 XMLSpy 製品機能と Altova 製品 117

4 編集ビュー 118

4.1 ファイルの自動的なバックアップ 120
4.2 テキストビュー 122
4.2.1 テキストビューでのフォーマット 123
4.2.2 ドキュメントの表示 124
4.2.3 テキストビューでの編集 127
4.2.4 ドキュメント内の移動 129
4.2.5 検索および置換 132
4.2.6 テキストビューでの入力ヘルパー 137
4.2.7 分割ビュー 138
4.2.8 テキストビューショートカット 140
4.3 グリッドビュー 141
4.3.1 グリッドビューでの編集 141
4.3.2 グリッドビューテーブル 143
4.3.3 グリッドビューでの入力ヘルパー 147
4.3.4 グリッドビューショートカット 148
4.4 スキーマ ビュー 151
4.4.1 XSD モード: XSD 1.0 と 1.1 153
4.4.2 スキーマ概要 156
4.4.3 コンテンツ モデル ビュー 168
4.4.4 属性、アサーション、同一性制約 187
4.4.5 スキーマ ビューにおける入力ヘルパー 201
4.4.6 検証とスマートフィックス 211
4.4.7 アサーションメッセージ 212
4.4.8 基底型の修正 214
4.4.9 スマート制約 215
4.4.10 xml:base、xml:id、xml:lang、xml:space 220
4.4.11 戻ると進む: 位置の移動 222
4.5 Authentic ビュー 223
4.6 ブラウザービュー 224
4.7 アーカイブビュー 225
4.8 共通のショートカット 227

5 XML 229

5.1 XML ドキュメントを作成、開く、保存する 230
5.2 スキーマの割り当てと検証 232
5.3 テキストビュー内の XML 234
5.4 グリッドビュー内の XML 236
5.4.1 ドキュメントの表示 238
5.4.2 ドキュメント構造 241
5.4.3 ドキュメントコンテキスト 243
5.4.4 グリッドビュー内でのドラッグとドロップ 246
5.4.5 グリッドビュー内の入力ヘルパー 249
5.4.6 テーブルの表示 250
5.4.7 フィルター 254
5.4.8 フォーミュラ 255
5.4.9 イメージ 258
5.4.10 チャート 259
5.4.11 グリッドビュー内のコンテキストメニュー 262
5.4.12 グリッドビュー設定 265
5.5 Authentic ビュー内の XML 268
5.6 XML ドキュメントの入力ヘルパー 270
5.7 XML ドキュメントの検証 271
5.8 XML フラグメントの挿入 273
5.9 XSLT と XQuery による処理 275
5.10 PDF フォント 277
5.11 その他の機能 279

6 DTD と XML スキーマ 281

6.1 DTD 282
6.2 XML スキーマ 284
6.3 スキーマサブセット 285
6.4 SchemaAgent との連携 289
6.4.1 SchemaAgent Server へ接続 290
6.4.2 検索パス上のスキーマを開く 292

6.4.3 IIR の使用 293
6.4.4 SchemaAgent でスキーマを閲覧 298
6.4.5 SchemaAgent 検証 298
6.5 XMLSpy 内のカタログ 301
6.5.1 カタログのしくみ 301
6.5.2 XMLSpy 内のカタログ構造 302
6.5.3 カタログのカスタム化 303
6.5.4 環境変数 305
6.6 スキーマの中を検索 307
6.6.1 検索語 309
6.6.2 コンポーネント 310
6.6.3 プロパティ 311
6.6.4 スコープ 314
6.6.5 検索と置き換えコマンド 315
6.6.6 結果と情報 317
6.6.7 グローバルコンポーネントの検索と名前変更 317

7 XSLT 320
7.1 XSLT ドキュメント 321
7.2 XSLT 処理 323
7.3 XSL アウトライン 325
7.3.1 XSL アウトラインウィンドウ 326
7.3.2 情報ウィンドウ 329

8 XQuery 332
8.1 XQuery ドキュメントの編集 334
8.1.1 XQuery ドキュメント 334
8.1.2 XQuery 入力ヘルパー 335
8.1.3 XQuery の構文による色分け 336
8.1.4 XQuery インテリジェント編集 338
8.2 XQuery 評価 341
8.3 XQuery 検証 342
8.4 XQuery/Update 実行 343

8.5 XQuery Update Facility ... 345
8.5.1 アップデートのプレビューと適用 ... 345
8.5.2 Update オペレーションと構文 ... 347
8.6 XQuery と XML データベース ... 351

9 XSLT/XQuery デバッガー とプロファイラー 355

9.1 XSLT と XQuery デバッガー ... 356
9.1.1 メカニズムとインターフェース ... 357
9.1.2 コマンドとツールバーアイコン ... 358
9.1.3 デバッガー設定 ... 362
9.1.4 デバッグセッションの開始 ... 363
9.1.5 情報ウィンドウ ... 364
9.1.6 ブレークポイント ... 372
9.1.7 トレースポイント ... 375
9.1.8 デバッガーショートカット ... 380

10 XPath/XQuery 条件式 381

10.1 XPath/XQuery ウィンドウについて ... 382
10.2 式の評価 ... 384
10.3 式のデバッグ ... 389
10.4 式ビルダー ... 395
10.5 JSONX のための Query 式 ... 397
10.6 注意点 ... 399

11 Authentic 401

11.1 Authentic View チュートリアル ... 403
11.1.1 AUTH-VIEW 内で XML ドキュメントを開く ... 404
11.1.2 Authentic View インターフェイス ... 405
11.1.3 ノードを操作する ... 407
11.1.4 Authentic View 内にデータを挿入する ... 410
11.1.5 属性の値を入力する ... 411
11.1.6 エンティティを入力する ... 412
11.1.7 ドキュメントを印刷する ... 413

11.2 Authentic View インターフェイス ... 415
11.2.1 GUI の概要 ... 415
11.2.2 Authentic View ツールバーアイコン ... 416
11.2.3 Authentic View メインウィンドウ ... 419
11.2.4 Authentic View エントリヘルパー ... 421
11.2.5 Authentic View コンテキストメニュー ... 425
11.3 Authentic View 内で編集する ... 428
11.3.1 基本的な編集 ... 428
11.3.2 Authentic View 内のテーブル ... 432
11.3.3 DB の編集 ... 440
11.3.4 日付と作業する ... 446
11.3.5 エンティティを定義する ... 448
11.3.6 XML 署名 ... 449
11.3.7 Authentic View 内のイメージ ... 450
11.3.8 Authentic View 内のキーストロール ... 451
11.4 Authentic スクリプト ... 452

12 HTML と CSS 454
12.1 HTML ... 455
12.2 CSS ... 457

13 JSON と JSON スキーマ 462
13.1 JSON データ ... 464
13.2 JSON スキーマ ... 467
13.3 JSON ラインと JSON コメント ... 469
13.4 JSON テキストビュー ... 470
13.5 JSON グリッドビュー ... 475
13.5.1 グリッドビュー内でのドキュメントの表示 ... 478
13.5.2 ドキュメント構成の編集 ... 481
13.5.3 ドキュメントコンテンツの編集 ... 482
13.5.4 グリッドビュー内でのドラッグとドロップ ... 485
13.5.5 テーブルの表示 ... 488
13.5.6 フィルター ... 492

13.5.7 フォーミュラ 493
13.5.8 イメージ 497
13.5.9 チャート 498
13.5.10 グリッドビュー内のコンテキストメニュー 500
13.5.11 グリッドビューの設定 502
13.5.12 JSON スキーマビュー 505
13.6 JSON ドキュメントの検証 537
13.7 JSON フラグメントの挿入 539
13.8 XSLT/XQuery を使用した JSON 変換 541
13.9 JSON のための XQuery 式 543
13.10 JSON インスタンスからの JSON スキーマの生成 545
13.11 JSON インスタンスを JSON スキーマから生成する 548
13.12 JSON と XML の間の変換 549

14 HTTP 550
14.1 リクエストの送信 551
14.2 送信するリクエストのインポート 555
14.3 レスポンスの受信 557

15 Office Open XML、ZIP、EPUB 560
15.1 OOXML ファイルの編集 562
15.2 OOXML サンプルファイル 564
15.3 ZIP ファイル 566
15.4 EPUB ファイル 568

16 データベース 572
16.1 データソースへの接続 573
16.1.1 接続ウィザードの開始 574
16.1.2 データベースドライバー概要 575
16.1.3 ADO 接続の設定 578
16.1.4 ADO.NET 接続のセットアップ 585
16.1.5 ODBC 接続の設定 591
16.1.6 JDBC 接続の設定 594

16.1.7 PostgreSQL 接続のセットアップ ... 598
16.1.8 SQLite 接続の設定 ... 599
16.1.9 グローバルリソースからの接続の使用 ... 602
16.1.10 データベース接続のサンプル ... 603
16.2 サポートされているデータベース ... 659

17 Altova グローバルリソース 660

17.1 グローバル リソースの定義 ... 661
17.1.1 ファイル ... 663
17.1.2 フォルダー ... 667
17.1.3 データベース ... 668
17.2 グローバルリソースを使用する ... 670
17.2.1 ファイルとフォルダーの割り当て ... 670
17.2.2 データベースの割り当て ... 673
17.2.3 構成の切り替え ... 674

18 プロジェクト 676

18.1 プロジェクトの作成と編集 ... 677
18.2 プロジェクトを使用する ... 681

19 RaptorXML Server 683

19.1 サーバーとサーバー構成の追加 ... 684
19.2 RaptorXML Server を使用して検証 ... 688
19.3 検証オプション ... 689
19.3.1 共通のオプション ... 689
19.3.2 DTD に対して XML ... 690
19.3.3 DTD ... 690
19.3.4 W3C スキーマに対して XML ... 691
19.3.5 W3C スキーマ ... 691
19.3.6 XBRL インスタンス ... 692
19.3.7 XBRL タクソノミ ... 693
19.3.8 XSLT ... 694
19.3.9 XQuery ... 695

19.3.10 JSON ... 696
19.3.11 JSON スキーマ ... 697
19.3.12 EDGAR ... 697
19.4 RaptorXML Server使用しての XSLT と XQuery ... 699

20 ファイル/ディレクトリの比較 701

20.1 ファイルの比較 ... 702
20.2 ディレクトリの比較 ... 703

21 Source Control 704

21.1 Setting Up Source Control ... 706
21.2 Supported Source Control Systems ... 707
21.3 Local Workspace Folder ... 709
21.4 Application Project ... 710
21.5 Add to Source Control ... 712
21.6 Working with Source Control ... 714
21.6.1 Add to, Remove from Source Control ... 714
21.6.2 Check Out, Check In ... 715
21.6.3 Getting Files as Read-Only ... 717
21.6.4 Copying and Sharing from Source Control ... 719
21.6.5 Changing Source Control ... 722
21.7 Source Control with Git ... 724
21.7.1 Enabling Git Source Control with GIT SCC Plug-in ... 725
21.7.2 Adding a Project to Git Source Control ... 725
21.7.3 Cloning a Project from Git Source Control ... 726

22 Visual Studio 内の XMLSpy 729

22.1 XMLSpy プラグインのインストール ... 730
22.2 XMLSpy スタンドアロンとの違い ... 731
22.3 Visual Studio での XMLSpy デバッガー ... 733

23 Eclipse 内の XMLSpy 734

23.1 Eclipse 用の XMLSpy プラグインをインストール ... 735
23.2 Eclipse 内の XMLSpy エントリポイント ... 737
23.3 XMLSpy のデバッガーパースペクティブ ... 740

24 メニューコマンド 741

24.1 ファイルメニュー ... 742
24.1.1 新規作成 ... 742
24.1.2 開く ... 746
24.1.3 再ロード ... 751
24.1.4 エンコード ... 751
24.1.5 閉じる、全て閉じる、非アクティブを全て閉じる ... 752
24.1.6 保存、名前を付けて保存、全て保存 ... 752
24.1.7 メールで送信 ... 757
24.1.8 印刷 ... 758
24.1.9 印刷プレビュー、印刷設定 ... 760
24.1.10 最近使用されたファイル、終了 ... 760
24.2 編集メニュー ... 762
24.2.1 元に戻す、やり直し ... 763
24.2.2 切り取り、コピー、貼り付け、削除 ... 763
24.2.3 XML テキストとしてコピー ... 764
24.2.4 構造化テキストとしてコピー ... 764
24.2.5 XPath をコピー ... 768
24.2.6 XPointer をコピー ... 768
24.2.7 挿入 ... 768
24.2.8 整形 ... 771
24.2.9 空白の削除 ... 772
24.2.10 全て選択 ... 772
24.2.11 検索、次を検索 ... 773
24.2.12 置換 ... 776
24.2.13 ファイル内の検索 ... 778
24.2.14 ブックマーク コマンド ... 780
24.2.15 コメントの IN/OUT ... 781
24.3 プロジェクトメニュー ... 782
24.3.1 新規プロジェクト ... 784

24.3.2 プロジェクトを開く 784
24.3.3 プロジェクトの再ロード 785
24.3.4 プロジェクトを閉じる 785
24.3.5 プロジェクトの保存、名前を付けて保存 785
24.3.6 ソース管理 785
24.3.7 プロジェクトにファイルを追加 799
24.3.8 プロジェクトにグローバルリソースを追加 799
24.3.9 プロジェクトに URL を追加 799
24.3.10 プロジェクトにアクティブなファイルを追加 799
24.3.11 プロジェクトにアクティブならびに関係するファイルを追加 800
24.3.12 プロジェクトにアクティブならびに関係するフォルダーを追加 800
24.3.13 プロジェクトに外部フォルダーを追加 800
24.3.14 プロジェクトに外部ウェブフォルダーを追加 803
24.3.15 スクリプトの設定 808
24.3.16 プロパティ 808
24.3.17 最近使用されたプロジェクト 811
24.4 XML メニュー 812
24.4.1 挿入 812
24.4.2 追加 820
24.4.3 子の追加 827
24.4.4 変換 834
24.4.5 テーブル 836
24.4.6 左へ移動 838
24.4.7 右へ移動 838
24.4.8 要素の中へ 838
24.4.9 XPath を評価 839
24.4.10 整形式のチェック 839
24.4.11 XML の検証 841
24.4.12 サーバー上の XML の検証(ハイパフォーマンス) 845
24.4.13 WSDL ファイルの検証 846
24.4.14 編集中に検証 847
24.4.15 入力ヘルパーの更新 847
24.4.16 名前空間プレフィックス 847
24.5 JSON メニュー 848
24.5.1 型 848

24.5.2 子を前/後に挿入、子の付加、子の追加 849
24.5.3 配列/オブジェクト内のラップ 849
24.5.4 右から左への読み取り 849
24.5.5 移動 850
24.5.6 コピー 850
24.5.7 展開、折りたたみ 850
24.5.8 昇順に並べ替え/降順に並べ替え 851
24.6 DTD/スキーマメニュー 852
24.6.1 DTD の割り当て 852
24.6.2 スキーマの割り当て 853
24.6.3 他の DTD を読み込み 853
24.6.4 DTD へ移動 853
24.6.5 スキーマへ移動 854
24.6.6 定義へ移動 854
24.6.7 DTD/スキーマの生成 854
24.6.8 DTD のフラット化 856
24.6.9 DTD をスキーマに変換 856
24.6.10 フラット化されたスキーマ 858
24.6.11 スキーマを DTD に変換する 858
24.6.12 UML へ変換 859
24.6.13 MapForce で DB/Excel/EDI から XML を生成 860
24.6.14 StyleVision で HTML/PDF/Word 出力をデザイン 860
24.6.15 サンプル XML/JSON ファイルの生成 860
24.6.16 メモリキャッシュの破棄 864
24.7 スキーマ設計メニュー 865
24.7.1 スキーマ設定 865
24.7.2 図の保存 868
24.7.3 ドキュメンテーションの生成 868
24.7.4 表示設定 873
24.7.5 ズーム 876
24.7.6 全てのグローバルを表示 877
24.7.7 図を表示 877
24.7.8 データベースに対するスキーマ拡張 877
24.7.9 SchemaAgent Server に接続 881
24.7.10 SchemaAgent Server から切断 882

24.7.11 SchemaAgent で表示 882
24.7.12 SchemaAgent 検証 882
24.7.13 スキーマサブセットの作成 883
24.7.14 スキーマのフラット化 884
24.8 XSL/XQuery メニュー 886
24.8.1 XSL 変換 888
24.8.2 XSL-FO 変換 888
24.8.3 XSL パラメーター / XQuery 実行 890
24.8.4 XQuery/Update 実行 893
24.8.5 XSL の割り当て 894
24.8.6 XSL-FO の割り当て 894
24.8.7 サンプル XML ファイルの割り当て 894
24.8.8 XSL へ移動 895
24.8.9 デバッガーの開始/実行 895
24.8.10 デバッガーの停止 895
24.8.11 デバッガーの再起動 896
24.8.12 デバッグセッションの終了 896
24.8.13 ステップイン 896
24.8.14 ステップアウト 896
24.8.15 ステップオーバー 897
24.8.16 現在実行中のノードを表示 897
24.8.17 ブレークポイントの挿入/削除 897
24.8.18 トレースポイントの挿入/削除 897
24.8.19 ブレークポイントの有効化/無効化 898
24.8.20 トレースポイントの有効化/無効化 898
24.8.21 ブレークポイント/トレースポイント 898
24.8.22 デバッグウィンドウ 899
24.8.23 デバッグ設定 899
24.9 Authentic メニュー 901
24.9.1 新規ドキュメント 901
24.9.2 データベースの編集 902
24.9.3 StyleVision スタイルシートの割り当て 903
24.9.4 StyleVision スタイルシートの編集 904
24.9.5 XML データの新たな行を選択し編集 904
24.9.6 XML 署名 905

24.9.7 XML エンティティの定義 906
24.9.8 マークアップの表示 908
24.9.9 リッチエディット 908
24.9.10 行の追加/挿入/複製/削除 909
24.9.11 上に/下に移動 910
24.9.12 HTML、RTF、PDF、Word 2007+ ドキュメントの生成 910
24.9.13 信頼された場所 910
24.10 DB メニュー 912
24.10.1 データベースをクエリする 912
24.10.2 IBM DB2 925
24.10.3 SQL Server 931
24.10.4 Oracle XML DB 933
24.11 変換メニュー 938
24.11.1 テキストファイルのインポート 938
24.11.2 データベースのデータをインポート 941
24.11.3 Microsoft Word ドキュメントをインポート 946
24.11.4 データベース構造から XML スキーマを作成 946
24.11.5 XML スキーマを元にデータベースをインポート 950
24.11.6 XML スキーマを元にデータベース構造を作成 951
24.11.7 テキストファイルへエクスポート 954
24.11.8 データベースへエクスポート 957
24.11.9 XML インスタンス を JSON に変換する（ JSON から XML インスタンスに変換する） 959
24.11.10 XML スキーマ を JSON スキーマに変換する（ JSON スキーマから XML スキーマに変換する） 962
24.12 表示メニュー 965
24.12.1 テキストビュー 965
24.12.2 拡張グリッドビュー 966
24.12.3 スキーマデザインビュー 966
24.12.4 Authentic View 966
24.12.5 ブラウザービュー 967
24.12.6 展開 967
24.12.7 縮退 967
24.12.8 全て展開 968
24.12.9 未選択を縮退 968
24.12.10 最適な幅 968

24.12.11 ワードラップ 968
24.12.12 行/文字へ移動 969
24.12.13 ファイルへ移動 969
24.12.14 テキストビュー設定 969
24.13 ブラウザーメニュー 972
24.13.1 戻る 972
24.13.2 進む 972
24.13.3 中止 973
24.13.4 最新の状態に更新 973
24.13.5 フォント 973
24.13.6 別のウィンドウ 973
24.14 ツールメニュー 974
24.14.1 スペリング 974
24.14.2 スペルチェックのオプション 978
24.14.3 スクリプトエディター 982
24.14.4 マクロ 983
24.14.5 比較 983
24.14.6 グローバルリソース 993
24.14.7 アクティブな構成 994
24.14.8 Raptor Server の管理 994
24.14.9 Raptor Server と構成 998
24.14.10 カスタマイズ 998
24.14.11 ツールバーとウィンドウの復元 1013
24.14.12 オプション 1014
24.15 ウィンドウメニュー 1044
24.15.1 重ねて表示 1044
24.15.2 上下に並べて表示 1044
24.15.3 左右に並べて表示 1044
24.15.4 プロジェクトウィンドウ 1044
24.15.5 情報ウィンドウ 1045
24.15.6 入力ヘルパー 1045
24.15.7 出力ウィンドウ 1045
24.15.8 プロジェクトおよび入力ヘルパー 1045
24.15.9 全てオン/オフ 1045
24.15.10 現在開かれているウィンドウのリスト 1046

24.16..ヘルプメニュー..... 1047
24.16.1 目次、インデックス、検索..... 1047
24.16.2 キーボードマップ..... 1048
24.16.3 ライセンス登録、注文フォーム、登録、最新情報のチェック..... 1049
24.16.4 他のコマンド..... 1052
24.17..コマンドライン..... 1053

25 プログラマーのレファレンス 1054

25.1 ...Scripting Editor..... 1056
25.1.1 Creating a Scripting Project..... 1057
25.1.2 Built-in Commands..... 1069
25.1.3 Enabling Scripts and Macros..... 1079
25.2 ...IDE Plugins..... 1081
25.2.1 Registration of IDE PlugIns..... 1081
25.2.2 ActiveX Controls..... 1082
25.2.3 Configuration XML..... 1082
25.2.4 ATL Sample Files..... 1085
25.2.5 IXMLSpyPlugIn..... 1090
25.3 ...Application API..... 1095
25.3.1 Overview..... 1096
25.3.2 Interfaces..... 1130
25.3.3 Interfaces (obsolete)..... 1321
25.3.4 Enumerations..... 1358
25.3.5 Application API for Java (obsolete)..... 1372
25.4 ...ActiveX Integration..... 1413
25.4.1 必要条件..... 1413
25.4.2 ActiveX コントロールをツールボックスに追加する..... 1414
25.4.3 アプリケーションレベルの統合..... 1416
25.4.4 ドキュメントレベルの統合..... 1418
25.4.5 ActiveX 統合のサンプル..... 1421
25.4.6 コマンド レファレンス..... 1440
25.4.7 オブジェクトレファレンス..... 1459

26 付録 1481
26.1 ...XSLT および XQuery エンジンに関する情報 1482
26.1.1 XSLT 1.0 1482
26.1.2 XSLT 2.0 1482
26.1.3 XSLT 3.0 1484
26.1.4 XQuery 1.0 1484
26.1.5 XQuery 3.1 1487
26.2 ...XSLT と XPath/XQuery 関数 1488
26.2.1 Altova 拡張関数 1489
26.2.2 その他の拡張関数 1561
26.3 ...DB 内で生成された XML スキーマのデータ型 1578
26.3.1 ADO 1578
26.3.2 MS Access 1579
26.3.3 MS SQL Server 1579
26.3.4 MySQL 1580
26.3.5 ODBC 1581
26.3.6 Oracle 1582
26.3.7 Sybase 1582
26.4 ...XML スキーマから生成された DB のデータ型 1584
26.4.1 MS Access 1584
26.4.2 MS SQL Server 1585
26.4.3 MySQL 1587
26.4.4 Oracle 1589
26.5 ...技術データ 1591
26.5.1 OS とメモリ要件 1591
26.5.2 Altova XML バリデーター 1591
26.5.3 Altova XSLT と XQuery エンジン 1591
26.5.4 Unicode のサポート 1591
26.5.5 インターネットの使用 1592
26.6 ...ライセンス情報 1593
26.6.1 電子的なソフトウェアの配布 1593
26.6.2 ソフトウェアのアクティベーションとライセンスの計測 1593
26.6.3 エンドユーザー使用許諾契約書 1594

インデックス 1595

1 XMLSpy とこのドキュメントについて

Altova XMLSpy 2021 Professional Edition はJSON、XML、XML スキーマ、XSLT、とXQueryを含むデザイン、編集、デバッグのためのエンタープライズクラスアプリケーションのためのパワフルな XML とJSON エディターです。

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

このドキュメントは以下のセクションにより構成されています:

- 使用を開始するための チュートリアル 29
- 技術またはXMLSpy-固有の機能により分類されたXMLSpy 機能の説明
- メニューコマンド 741 の説明
- プログラマーのレファレンス 1054
- XMLSpy の(i) XSLT とXQuery エンジン、および(ii) Altova のXPath/XQuery 拡張関数に関する情報を含む付録 1481

最終更新日: *2021 年 03 月 26 日*

1.1 新規機能 2021

バージョン 2021 リリース 2

- XML ドキュメントのグリッドビュー[236] が多種の新規機能と共に拡張されました。
- XML ドキュメントのグリッドビュー[236] に関するコマンドと設定が簡単使用のために整理されました グリッドビュー設定[265] とオプションダイアログ[1022] を参照してください。
- Settings of the JSON ドキュメントのグリッドビュー[475] の設定が簡単使用のために整理されました グリッドビュー設定[502] とオプションダイアログ[1022] を参照してください。
- DTD ドキュメントのグリッドビュー[282] が新規機能と共に拡張されました。
- XML グリッドビュー[246] と JSON グリッドビュー[485] 内の新規のドラッグオーバーレイ機能はグリッドビュー内のドラッグアンドドロップの使用を促進する重要な情報を提供します。
- 新規機能により XML フラグメント[273] と JSON フラグメント[539] を外部ソースから簡単に素早く追加することができます。
- グリッドビュー内で（ XML コメントなどの）追加コンポーネントを コンポーネント固有の書式設定[1031] に与えることができます。各コンポーネント型はグリッドビュー内で割り当てられた書式設定で表示されます。例えば、コメントはグリッドビュー内で黄色でハイライトされています。
- 編集中にドキュメントを検証する[847] 新規機能を をオン/オフに切り替えることができます。
- Eclipse へのサポート[734] が拡張され、以下のバージョンがカバーされています: 2020-12; 2020-09; 2020-06; 2020-03.

バージョン 2021

- JSON メニュー[848] は以下の JSON グリッドビューの拡張機能を提供します: グリッドビューコンポーネントを選択し、アイテムを前後に挿入する。選択されたグリッドビューコンポーネントを 上/下、または、 左/右に移動する。ドキュメントからすべてのコメントを削除する。
- 新規の JSON グリッドビュー機能 (i) コンテキストメニュー[500] が簡素化されました。(ii) グリッドをズームすることができます[478]。(iii) 複数の兄弟コンポーネントを 展開/折りたたむ[478] ことができます。(iv) 文字列を含むセルは読み取りの方向を右から左に設定することができます（アラブ語およびヘブライ語など読み取りが右から左の言語と作業する際にこの機能は役に立ちます）。コピーコマンド[500] が拡張されました。
- JSON スキーマ検証[505] が `$id` プロパティを考慮するようになりました。
- 新規の スキーマ情報にアクセスするための Altova 拡張関数[1528]。
- Eclipse へのサポート[734] が拡張され、以下をカバーしています: 2020.06; 2020.03; 2019.12; 2019.09.

1.1.1 バージョン 2020

バージョン 2020 リリース 2

- 予期なくプログラムが終了した場合、編集中のファイルは 自動的にバックアップされ復元されます[120]。自動的にバックアップを実行し[1015] その頻度を設定することができます。
- 変更時に編集 機能はテキストビューと JSON グリッドビュー[482] 内での記入時に検証および整形式エラーをフラグします。この機能をプログラムオプション[1015] およびツールバーアイコンでオン/オフに切り替えることができます。
- スキーマビュー[151] 内では 多種のコンテキストメニュー内の型定義に移動[176] コマンドによりコンポーネントの型定義（単純型または複合型）に移動が可能になりました。
- 新規の JSON グリッドビューオプション により以下を行うことができます: (i) フィルターとフォーミュラ[492] がリクエストでアプリケーション全体のメタデータファイルに自動的に保存されるか、または保存されないかを決定することができます。(ii) この JSON メタデータファイルの場所を指定することができます。メタデータファイルを XMLSpy の複数のインスタンス全体の関連したファイル上に保管されているフィルターとフォーミュラに後で適用するために使用され、このためにフィルターとフォーミュラにポータビリティを提供します。

- JSON グリッドビュー[500]: コンテキストメニューは(i) 未選択のすべてのコンポーネントを折りたたむコマンド (ii) 任意で(コメントとして保管されている)フォーミュラを含むドキュメントから全てのコマンドを削除するコマンドを提供します。
- JSON グリッドビュー[475]: 改善された検索機能

バージョン 2020

- JSON グリッドビュー[475] は追加の編集機能を提供します: (I) 型の自動検知。(ii) セル内のコマンド。(iii) ビューの変更のためのXQuery (iv) JSON データからの追加出力の生成のためのXQuery フォーミュラ。
- JSON ラインとJSON コメント[469] のための検証と編集サポート
- JSON グリッドビュー内ではイメージを表示し[497] またチャートを作成し表示することができます[498]。
- オプションダイアログ[1014] (「ツール | オプション」) はテキストビュー内のJSON グリッドビュー とテキストビューの内のJSON ドキュメントの整形式 ための設定を提供します。
- XPath/XQuery ウィンドウ[381] の式エバリュエーターと式ビルダーは再デザインされ使用が簡単になりました。
- XPath/XQuery ウィンドウ[108] のXPath デバッガー[389] 機能に新規機能が搭載されました (i) ウォッチ式 (ii) より良い概要を表示する改善されたインターフェイス(iii) 更にパワフルな分析機能。この結果、XPath/XQuery 式をXML とJSON ファイルに対して更に簡単にテストしデバッグできるようになりました。
- 以前は異なるファイル型を持つファイルのデフォルトの編集ビューはオプションダイアログ[1018] 内で選択されていました。デフォルトの編集ビューは更に編集ビュー自身内で直接選択できるようになりました[118]。
- Visual Studio[729] 内でのXMLSpy の統合へのサポートがVisual Studio 2019 に拡張されました。
- Eclipse[734] 内でのXMLSpy の統合へのサポートが Eclipse 4.11 と4.12 に拡張されました。

1.1.2 バージョン 2019

バージョン 2019 リリース 3

XMLSpy **バージョン 2019r3** にて新たに追加された機能は以下のとおりです。

- 分割ビュー[138] で拡張されたテキストビュー。これにより、ドキュメントの他の部分を参照する単一ビューをスクロールする代わりに、長いドキュメントの異なる部分を横に並べる編集中のドキュメントに2つのビューを提供します。
- 新しいJSON スキーマバージョンへのサポート。draft-06 とdraft-07。ドキュメンテーションのJSON セクション[462]を参照してください。
- オプションダイアログ[1014] (「ツール | オプション」) は自動的に検知されたパスをオーバーライドするJava 仮想マシンを設定するオプションを提供します。
- Windows Server 2019 のためのサポート。
- Eclipse[734] へのサポートがEclipse 4.9 と4.10 に拡張されました。

バージョン 2019

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

- データベースへのサポートは以下のように拡張されました Firebird 3.0、IBM Informix 12.10、MariaDB 10.3、Microsoft SQL Server 2017、MySQL 8.0、PostgreSQL 10。
- Eclipse 4.8 との統合

1.1.3 バージョン 2018

バージョン 2018 リリース 2

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

- WADL ファイルからリクエストをインポートすることが可能なことに加え、HTTP 出力ウィンドウ[550] 内では、WSDL 1.1 または WSDL 2.0 ファイルから HTTP リクエストをインポート[555] することができます。
- JSON のための XPath/XQuery 3.1 のサポートが JSON 変換と JSON クエリ[541] 内で実装されました：(i) XPath/XQuery 出力ウィンドウ[108] が拡張され、JSON ドキュメントをクエリできるようになりました。(ii) XSLT 3.0 と XQuery 3.0 ドキュメントを使用して JSON ドキュメントが変換可能になりました[541]。(iii) XSLT と XQuery デバッガー[356] が拡張され JSON ファイルをデバッグできるようになりました。
- テキストビュー内の XSLT 3.0 値[321] のために XPath インテリジェントな編集のサポートが追加されました。
- テキストビューの編集が拡張され (i) 構文に関連下テキストの書式オプション[1029] (表示される空白など) が使用できるようになり、(ii) 多くの追加言語に対して適用できるようになりました。
- 次のファイル型のためのサポートの編集機能が追加されました CSS、Python、C-スタイル言語、オプションダイアログ[1014] のファイルタイプセクション[1018] 内で設定することのできる言語のファイル型への準拠。
- オプションダイアログ[1014] (「**ツール | オプション**」) がアプリケーションオプションをよりよく確認し、ユーザーフレンドリー性を向上するために編成されました。
- ネットワークプロキシ設定[1041] を使用することのできる複数のオプションを使用してカスタマイズできるようになりました。

バージョン 2018

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

- HTTP リクエストを作成し、ターゲット URL に送信し、レスポンスを受け取りチェックすることのできる新規 HTTP 出力ウィンドウ[550]。
- オプションダイアログ[1014] が再編成され、オプションの異なるグループがツリー構造に整理されました。これにより、異なるグループの概要がわかりやすくなりました。
- XBRL バージョン管理 1.0 のサポート
- Sybase 16、PostgreSQL 9.6、MySQL 5.7 のサポート
- Eclipse 4.7 との統合
- FOP 2.2 のサポート

1.1.4 バージョン 2017

バージョン 2017 リリース 3

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

- JSON5 のサポート[462]：この機能には自動補完、JSON スキーマに対する検証、XML からの変換、XML への変換が含まれます。
- JSON スキーマのサポートの拡張[507]：標準の JSON コンテナーにより名前がつけられた定義に加え、`resourceDefinitions` という名前のコンテナーからの定義が追加されました。
- テキストビュー：(XML、JSON、などの) 異なるドキュメントの種類の検証エラーは使用できる場合、番号付けられたマージン、スクロールバーにより示されており、エラーを素早くフィックスします。
- テキストビュー：ナビゲーション[129] と検索と置換[132] 関数が強化されました。

バージョン 2017

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

- テキストビュー：テキストのハイライト[129] は、現在選択されているテキストの一致を、一致に関するメタデータと共に表示します。
- テキストビュー：スクロールバー[131] は、ドキュメントのサイズ、ドキュメントのサイズに相対した現在の位置に関する情報を含んでいます。
- テキストビュー：検索と置き換え関数[132] への追加関数と簡単な使用方法

(C) 2015-2021 Altova GmbH

1.2 Windows ファイルパス

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

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

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

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

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

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

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

1.3 RaptorXML Server について

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

RaptorXML には2つのエディションがあります:

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

RaptorXML をコマンドラインから実行することができ、COM、Java、.NET, および Python のためのインターフェイスがあります。XMLSpy インターフェイス内から 683 Raptor サーバーを実行することもできます。

Altova Web サイト: XML 検証サーバー、XML バリデーター

2 XMLSpy チュートリアル

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

このチュートリアルは以下のように構成されます:

- XMLSpy インターフェイス 30。アプリケーションのグラフィカルユーザーインターフェイス(GUI) を詳しく理解することができます。
- XML スキーマの作成 41。XMLSpy の直感的なスキーマビューにて XML スキーマを作成する方法、ドラッグアンドドロップにより複雑なコンテンツモデルを作成する方法、そしてスキーマの設定方法を学習します。
- スキーマビューの機能を使用 56。複合型ならびに単純型、グローバルな要素参照、そして属性の列挙の作成方法を学習します。
- スキーマビューを使った スキーマのナビゲーション方法 68、スキーマのナビゲーション方法 68 を学習します。
- XML ドキュメントの作成 74。スキーマを XML ドキュメントに割り当て、グリッドビューやテキストビューを使ってXML ドキュメントを編集する方法、そして XMLSpy に組み込まれているバリデーターを使用して、XML ドキュメントを検証する方法を学習します。
- XSLT スタイルシートを使用して XML ファイルを変換する 94。XSLT ファイルを割り当て、XMLSpy に組み込まれたXSLT エンジンを使用することで変換を行います。
- XMLSpy プロジェクトにて作業する 98 ことにより、XML ドキュメントを簡単に整理することができます。

インストールと設定

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

チュートリアルファイル

チュートリアル ファイルはアプリケーション フォルダにあります:

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

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

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

2.1 XMLSpy インターフェイス

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

XMLSpy の開始

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

インターフェイスの概要

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

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

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

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

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

以下のビューを使用することができます: テキストビュー 122 、グリッドビュー 141 、スキーマビュー 151 、 Authentic View- ビュー 401 、アーカイブビュー 560 、およびブラウザービュー 224 。

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

2.1.1 表示

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

ドキュメントビューの切り替え

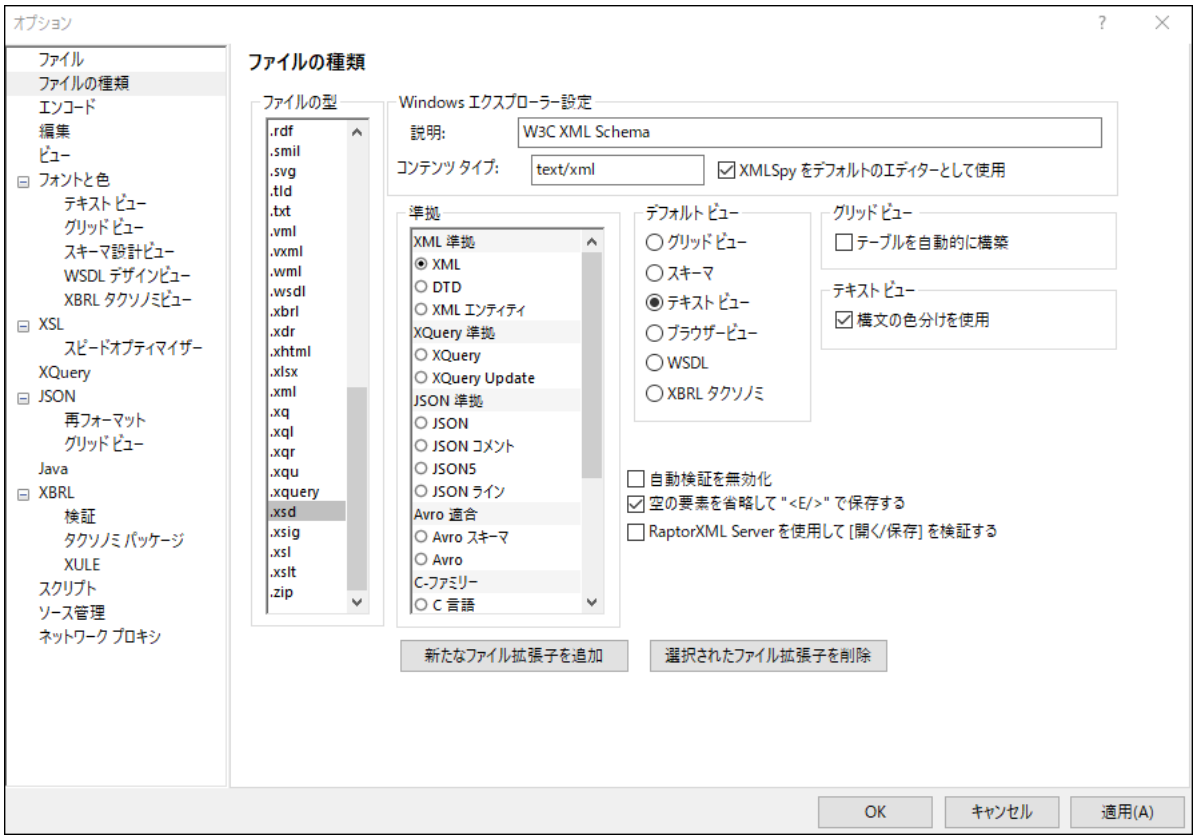
ドキュメントを開くと、そのドキュメントの型のために設定されたデフォルトのビューで開かれます。ドキュメントを以下の方法で開きます:

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

ドキュメントの型のデフォルトのビューの変更

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

1. **「ツール | オプション」**コマンドをクリックし、ファイルの種類のセクションに移動します（下のスクリーンショット参照。*XBRL と Avro サポートに関するオプションの一部は、Enterprise Edition のみでご使用いただけます*）。
2. 「ファイルの種類」ペインで、.xsd にスクロールして選択します（*スクリーンショットでハイライトされています*）。
3. 「デフォルトのビュー」ペインでテキストビューを選択します。



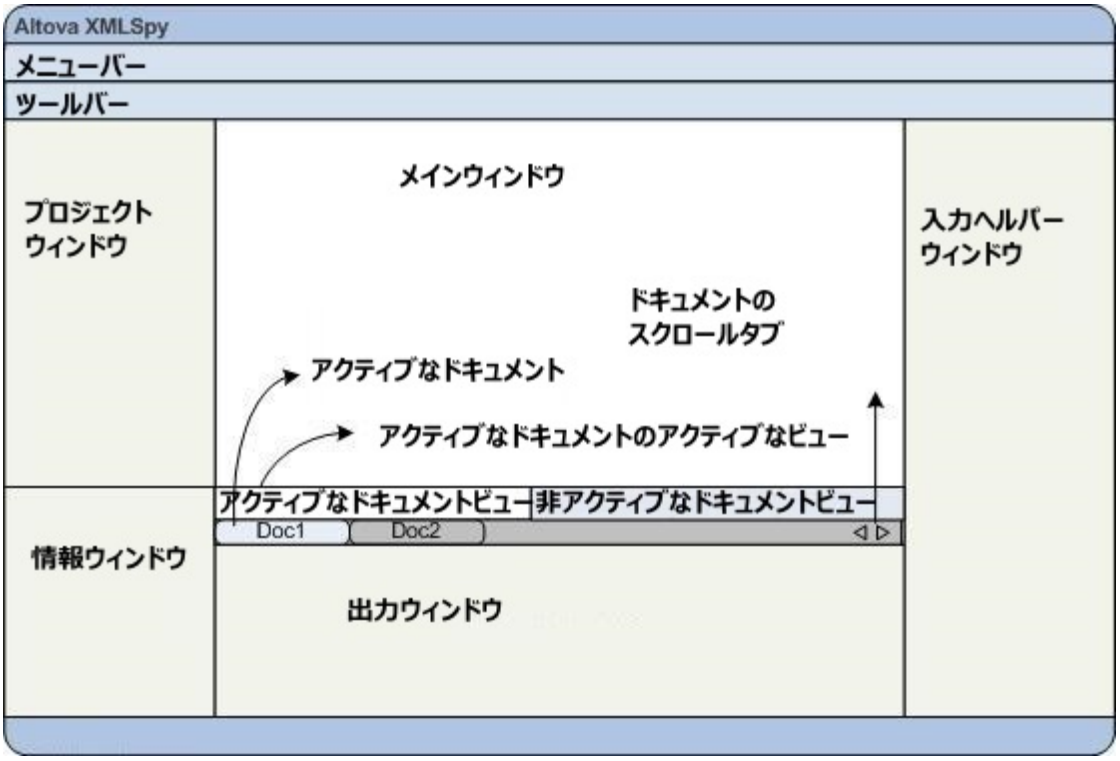
4. 「OK」をクリックします。
5. 「ファイル | 開く」コマンドをクリックして、`AddressFirst.xsd` ファイルを開きます。ファイルはテキストビューで開かれます。
6. スキーマビューに切り替えて、ファイルをこのビューで表示し、ファイルを「ファイル | 閉じる」を使用して閉じます。
7. 「ツール | オプション」を使用して、オプションダイアログに戻り、ファイル型のセクションで `.xsd` ファイルのデフォルトのビューをスキーマビューに戻します。

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

2.1.2 ウィンドウ

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

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



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

ウィンドウグループの表示のオンオフの切り替え

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

C:¥Documents and Settings¥<ユーザー名>¥マイ ドキュメント¥Altova¥XMLSpy2021\Examples\Tutorial フォルダー内の XML ファイルを開き、基本のコマンドの使用を練習し、コマンドの作動方法に慣れ親しんでください。ウィンドウグループの表示と非表示に関しての詳細は「XMLSpy インターフェイス」103 のセクションを参照してください。

保存された状態とバックアップされた状態

デフォルトでは XMLSpy は5秒の間隔で保存されていないドキュメントをバックアップします。メインウィンドウの下の各ファイルタブはファイルの保存/未保存の状態とバックアップの状態を表示するシンボルを使用して情報を提供します。作業中に定期的にこれらのインジケーターが表示されるためその意味を確認してください。これらのインジケーターの情報に関する情報に関してはファイルの自動的なバックアップ120 を参照してください。

スクリーン上でウィンドウを移動する

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

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

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

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

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

2.1.3 メニューとツールバー

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

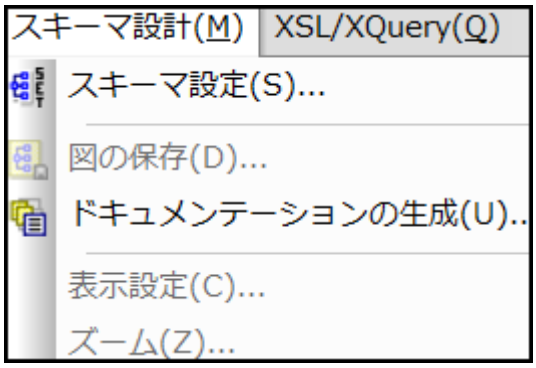
メニュー

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

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

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

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

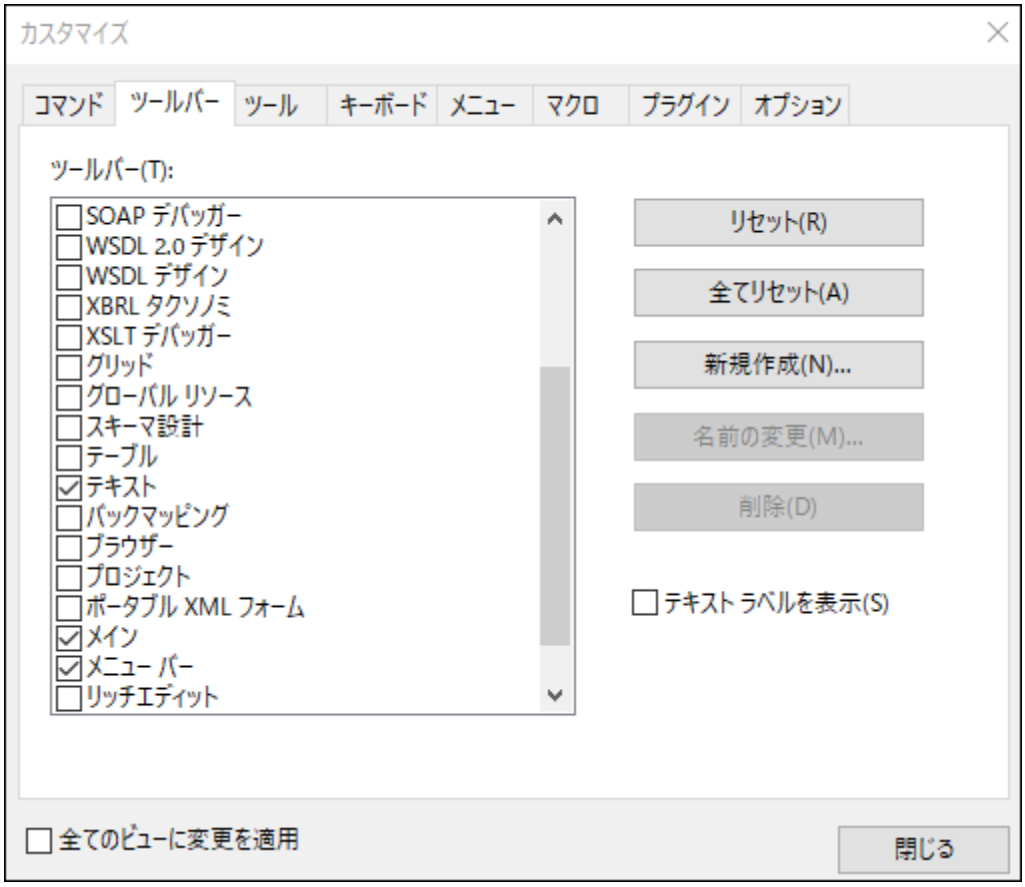


2. メニューの外側の空白をクリックして、メニューを消します。要素コンポーネントの左横にある **図の表示** アイコン をクリックします。これによりスキーマビューのコンポーネントモデルビューに移動します（スキーマビューの 2 番目のビュー; 1 番目のビューはスキーマ概要です）。スキーマデザインメニューを確認すると、「**図の保存**」、「**表示設定**」、および「**ズーム**」コマンドが有効化されていることが確認できます。これらのコマンドはスキーマビューのコンテンツモデルビュー内のみで有効化されますが、スキーマビューのスキーマ概要またはその他のビューでは有効化されません。 XML スキ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

全てのメニューコマンドの詳細に関しては、ユーザーマニュアルの「ユーザーレファレンス」のセクション 741を参照してください。

ツールバー

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



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

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

終了すると `AddressFirst.xsd` ファイルを閉じます。

2.1.4 テキストビューの設定

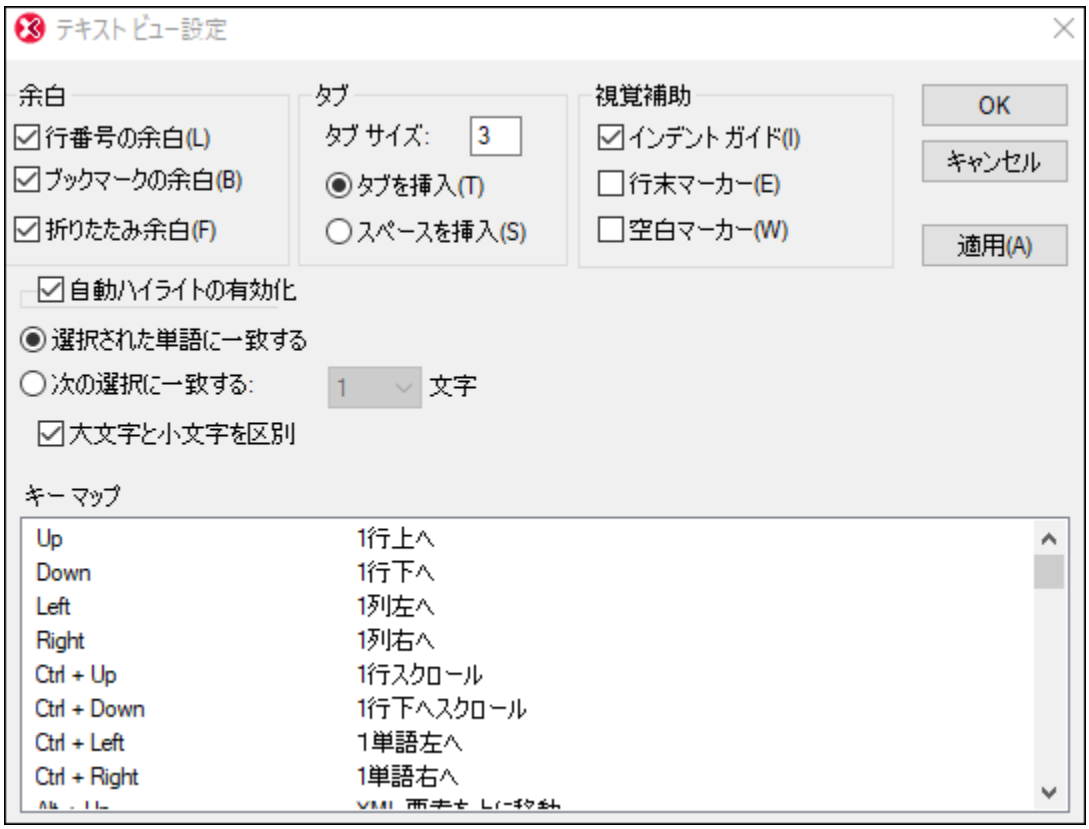
このセクションでは、再フォーマットされたドキュメントの設定および編集中のブックマークの使用方法を学習します。ドキュメントが再フォーマットされると、テキストビューで表示され、XML 階層の各下のレベルが前のレベルに比較してインデントされて表示されます (*下のスクリーンショット参照*)。ブックマークにより、ドキュメントの位置をマークし、戻る場所を素早く検索することができます。

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

整形出力

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

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



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

メモ オプションダイアログの整形出力セクション内で（「ツール | オプション | 整形出力」1022）、*インデントの使用* のチェックボックスのチェックを解除すると、すべての行はインデント無しに再フォーマットされます。

ブックマーク

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

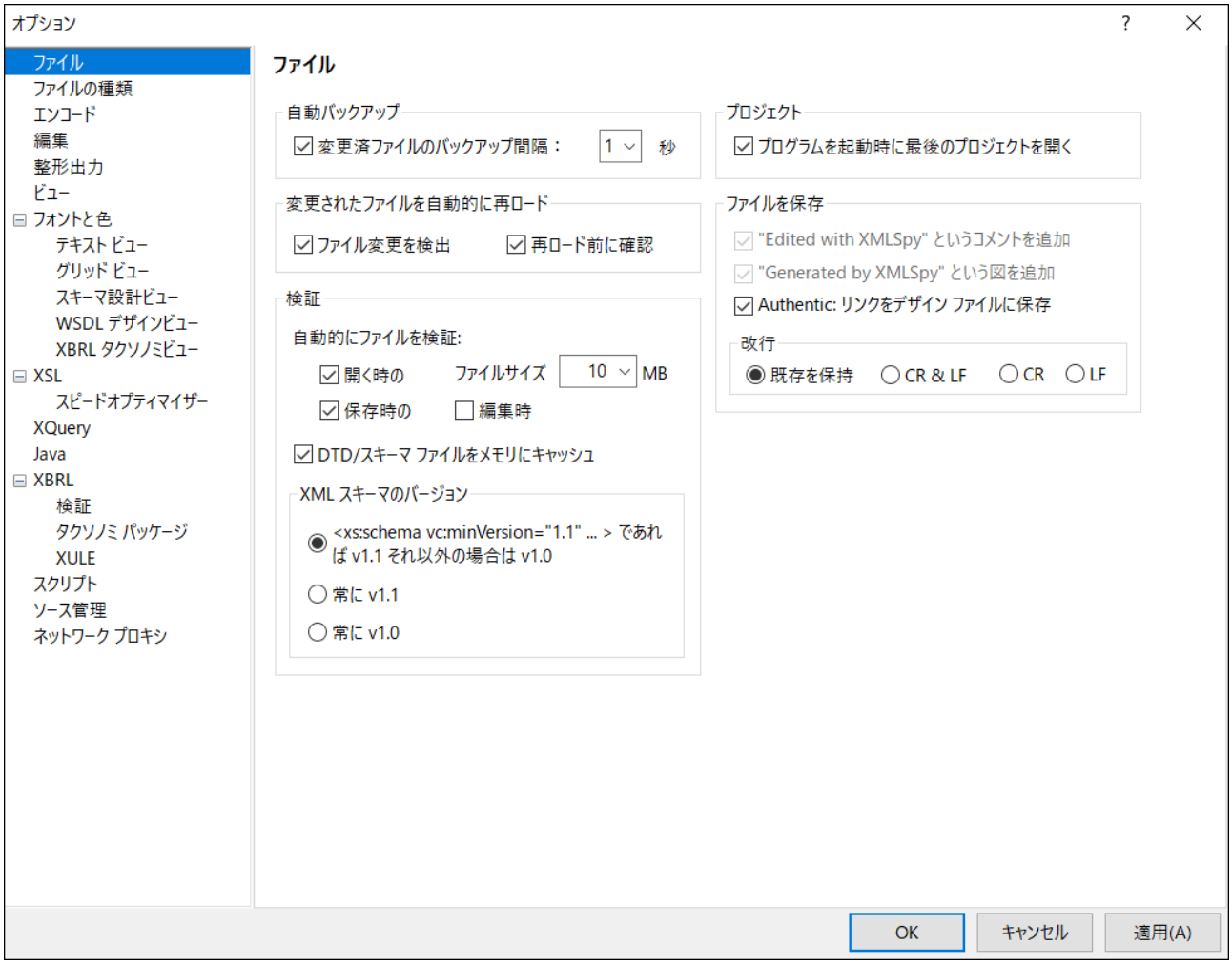
1. テキストビューの設定（「**表示 | テキストビューの設定**」*上のスクリーンショット*）で、ブックマージンのオプションがマージンペインで選択されていることを確認します。完了すると「OK」をクリックします。
2. ファイル`CompanyFirst.xml` のテキストビューで、ブックマークするラインにカーソルをポイントし、メニューコマンド「**編集 | ブックマークの挿入/削除**」を選択します。ラインはブックマークされ、ブックマークマージン内に青いブックマークで表示されます（*下のスクリーンショット参照*）。
3. ステップ 2 と同様に、他のラインのブックマークを作成します。

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

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

2.1.5 アプリケーションオプション

XMLSpy は機能が豊富なため、作業の多くのアスペクトを大幅に影響する多種のオプションを設定することができます。作業の必要条件と作業スタイルに適するようにアプリケーションの設定を変更し理解することは長期的にとても有利です。これらの設定はメニューコマンドで「ツール | オプション」1014を使用しオプションダイアログ内でアクセスすることができます（*下のスクリーンショット*）。



設定はセクションごとに整理されており、左側にリストされています。*XBRL オプションは Enterprise Edition のみで使用できることに注意してください*。各セクションを確認してどの設定が使用可能かを確かめてください。下のリスト内では、一般的に使用される機能に影響を与える設定の一部に着目してみました。全ての設定にの詳細に関してはオプションダイアログのドキュメンテーション[1014]に移動してください。設定の変更後「OK」クリックしてレジストリに変更を保存し、ダイアログを閉じます。「適用」ボタンにより現在開かれているドキュメントへの変更を表示することができます。

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

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

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

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

(C) 2015-2021 Altova GmbH

2.2 XML スキーマ：基本

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

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

目的

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

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

基本的なスキーマが完成したら、%XMLSPY%> によるXML スキーマのさらに高度な機能について次のセクション 56で学習します。その次には%XMLSPY%> のスキーマナビゲーションとドキュメンテーション 68に関するセクションが続きます。

このセクションで使用されるコマンド

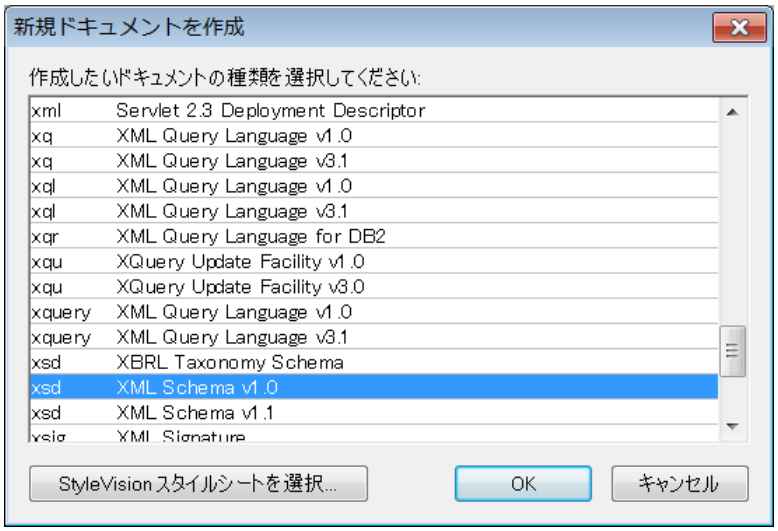
このセクションではスキーマビューだけを使用します。以下のコマンドが使われます：

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

2.2.1 新しい XML スキーマファイルの作成

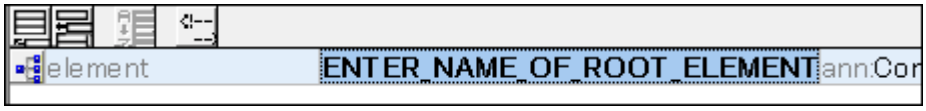
新しいXML スキーマファイルの作成：

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

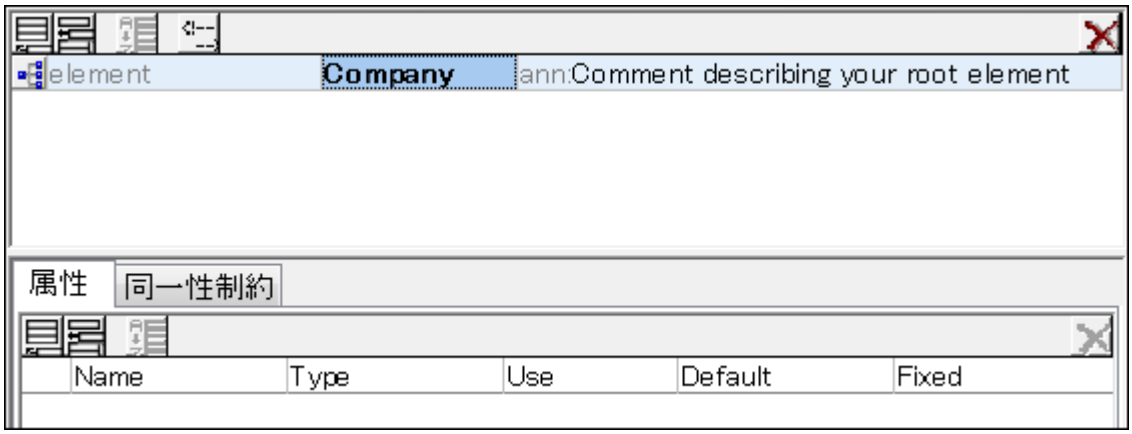


2. ダイアログから、 *XSD (XML Schema v1.0)* エントリ（ウィンドウ内のドキュメントの詳細とリストはスクリーンショットと異なる場合があります。）を選択し、「OK」で確定します。空のスキーマファイルがウィンドウのスキーマビューに表示されます。
3. スキーマ設計ツールバーから「XSD 1.0」モードボタン（*下のスクリーンショット参照*）をクリックして、スキーマビューがXSD 1.0編集モードになるように設定します。ます。

4. ルート要素を入力するよう求められます。



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



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

(C) 2015-2021 Altova GmbH

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

ファイルタブ内の色付けされたサークルのシンボルはファイルのバックアップの状態を示しています。これらのインジケーターの詳細に関してはファイルの自動的なバックアップ 120 を参照してください。

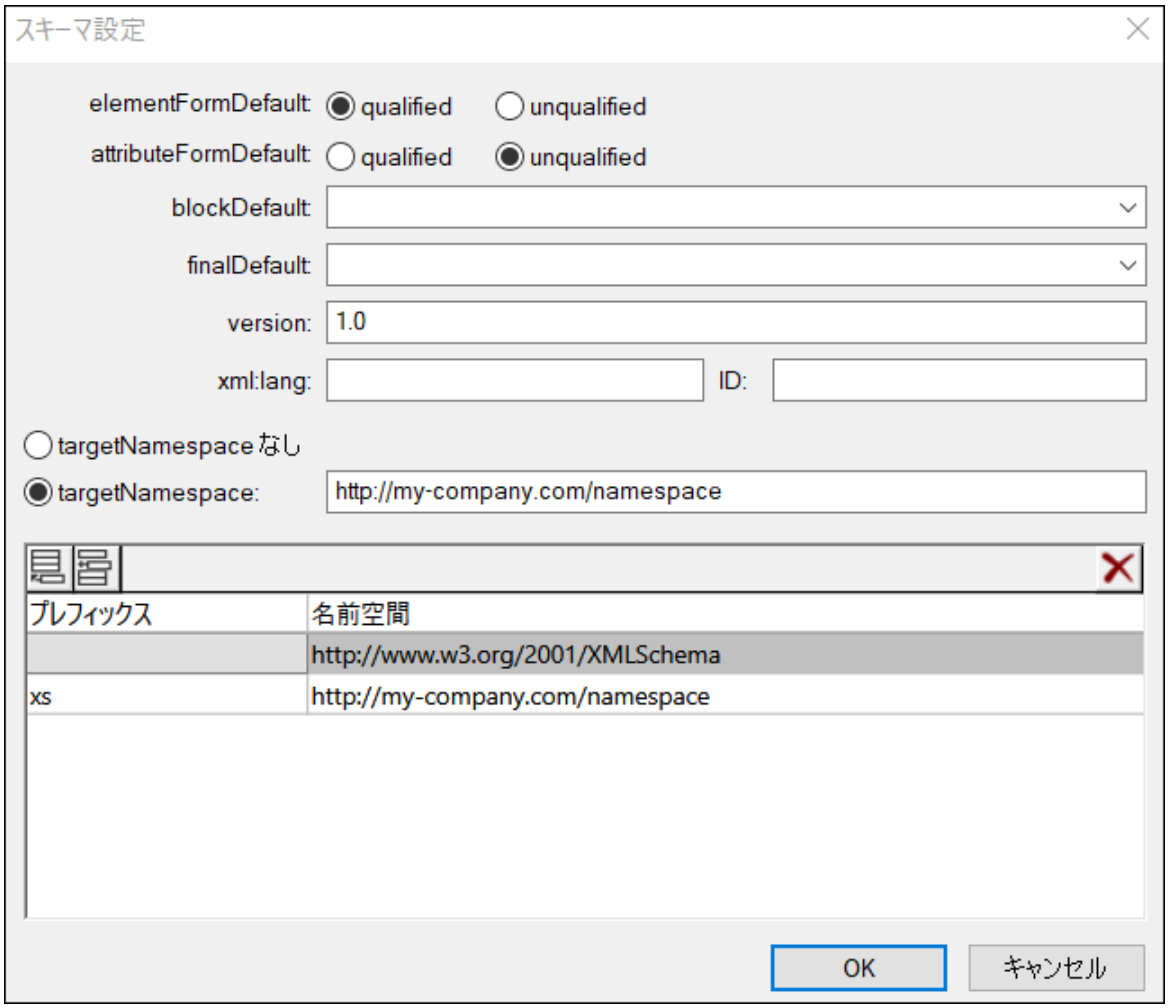
2.2.2 名前空間の定義

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

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

ターゲット名前空間の作成方法:

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



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

注意点:

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

2.2.3 コンテンツモデルの定義

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

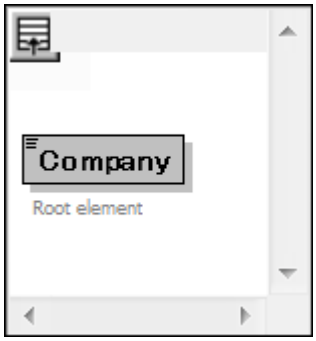
XMLSpy において、グローバルコンポーネントのコンテンツモデルは、スキーマビューのコンテンツモデルビュー内に表示されます。グローバルコンポーネントのコンテンツモデルを編集するには、グローバルコンポーネントの左にあるダイアグラム表示アイコン をクリックします。

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

基本的なコンテンツモデルの作成

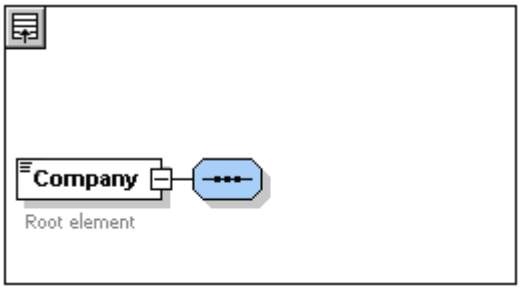
`Company` 要素のコンテンツモデルを作成するには:

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

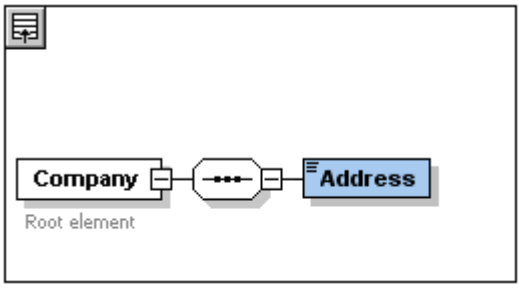


2. コンテンツモデルはコンポジッターとコンポーネントにより構成されます。コンポジッターは2つのコンポーネントの関係を指定します。`Company` コンテンツモデルでは、子要素を加えるために子コンポジッターを追加する必要があります。コンポジッターを追加するには、`Company` 要素を右クリックし、コンテキストメニューから「子を追加 | シーケンス」を選択します。（「シーケンス」、「選択」、そして「全て」は、コンテンツモデルで使用される3つのコンポジッターです）

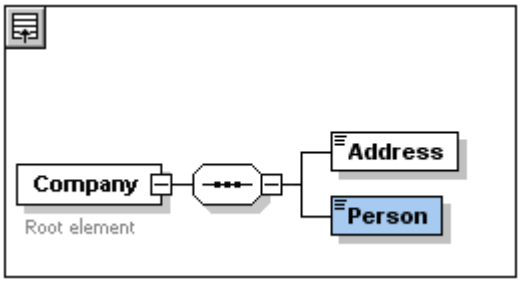
この操作によりシーケンスが指定され、そこから表示されるコンポーネントを定義するシーケンスコンポジッターが挿入されます。



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

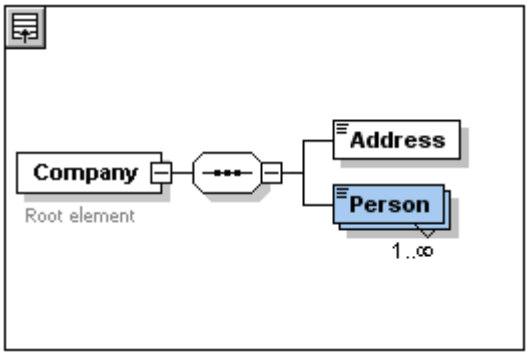


5. シーケンスコンポジッターを再度右クリックして、「子を追加 | 要素」を選択します。新たに作成された要素コンポジットの名前を Person にします。



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

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



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

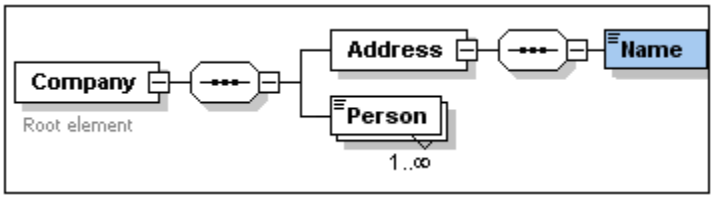
コンテンツモデル構造に新たな階層を追加

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

やり方:

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

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



複合型、単純型、スキーマデータ型

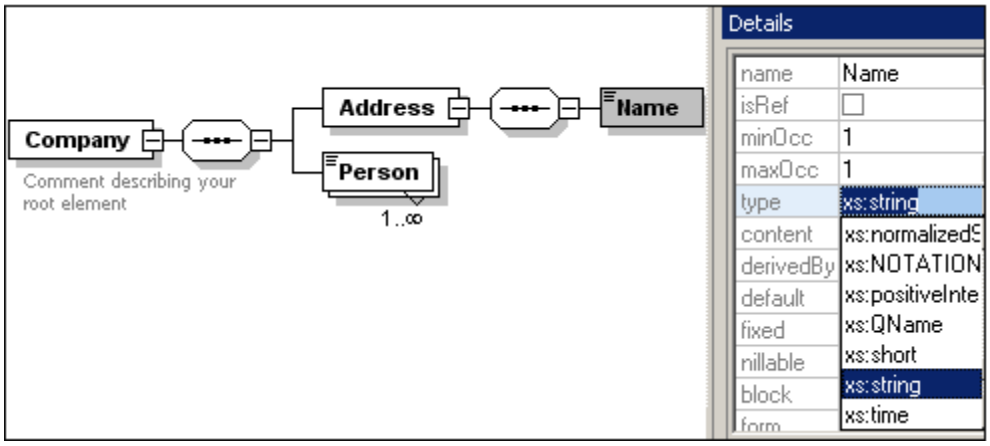
ここまで、要素の型を明示的に定義はしませんでした。テキストタブをクリックして、スキーマをテキストビューに表示させてください（下記参照）。シーケンスコンポジッターが挿入された場所には xs:complexType 内に xs:sequence 要素が挿入されていることが分かると思います。簡単に説明すると、Company そして Address 要素は子要素を持っているため複合型になります。複合型要素は属性または要素を持ちます。

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

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

Name 要素のデータ型を定義するには:

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



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

Name 要素のテキスト表記は以下のようになります:

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

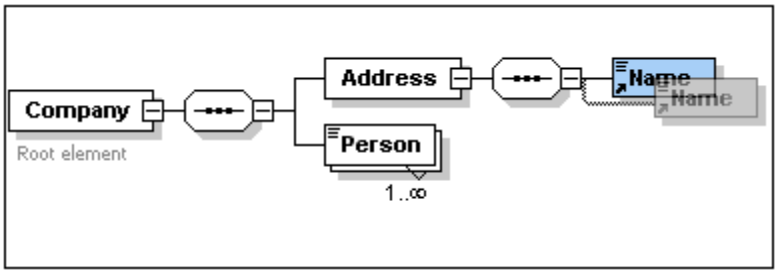
注意: 単純型要素は幾つかあるXMLスキーマデータ型のどれでも取ることができます。全ての場合において、要素ボックスにはテキストコンテンツであることを示すアイコンが表示されます。

2.2.4　ドラッグアンドドロップによる要素の追加

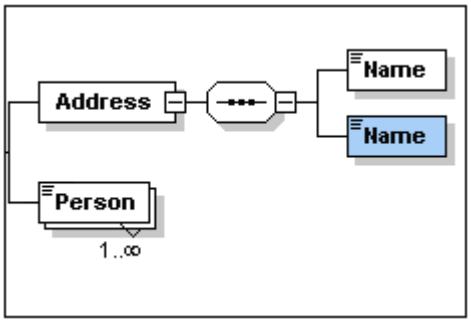
ここまでは、要素やコンポジッターを右クリックし、コンテキストメニューを使って要素を追加してきました。ドラッグアンドドロップを使うとコンテキストメニューより先素早く要素を追加することができます。このセクションでは、ドラッグアンドドロップを使って Address 要素にさらに子要素を追加して定義を完成させます。

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

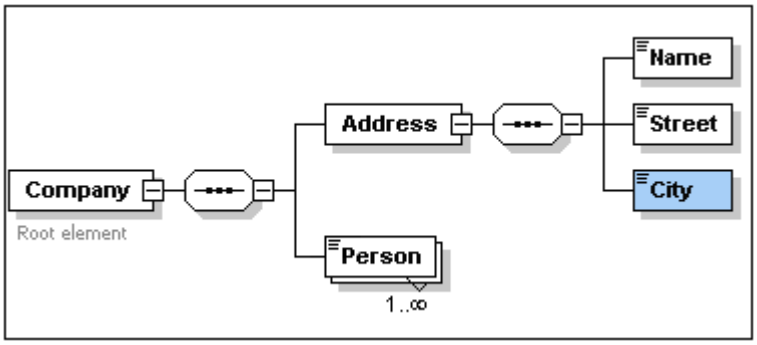
1. Address 要素のName 要素をクリックして、「Ctrl」キーを押下したまま要素ボックスをマウスでドラッグします。要素ボックスに表示される小さな「＋」アイコンはコピーが行われようとしていることを示しています。要素のコピーが接続線と一緒に表示され、要素がどこに作成されるのかを示しています。



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



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



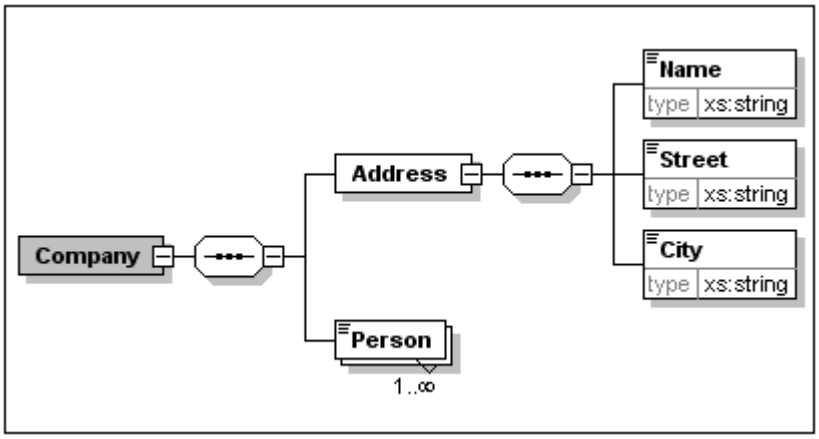
要素は `Name`、`Street`、そして `City` 要素からなるシーケンスを持つことになります。

2.2.5 コンテンツモデルビューの構成

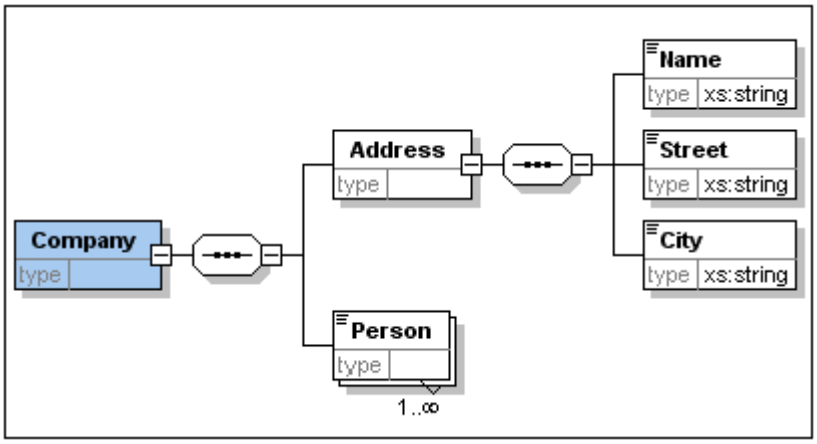
そろそろコンテンツモデルビューの設定を行っても良いでしょう。要素の型が要素ボックス内に表示されるようにビューを構成します。以下の手順を行ってください：

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

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



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



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

以下の点に注意してください：

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

2.2.6 基本的なスキーマの完成

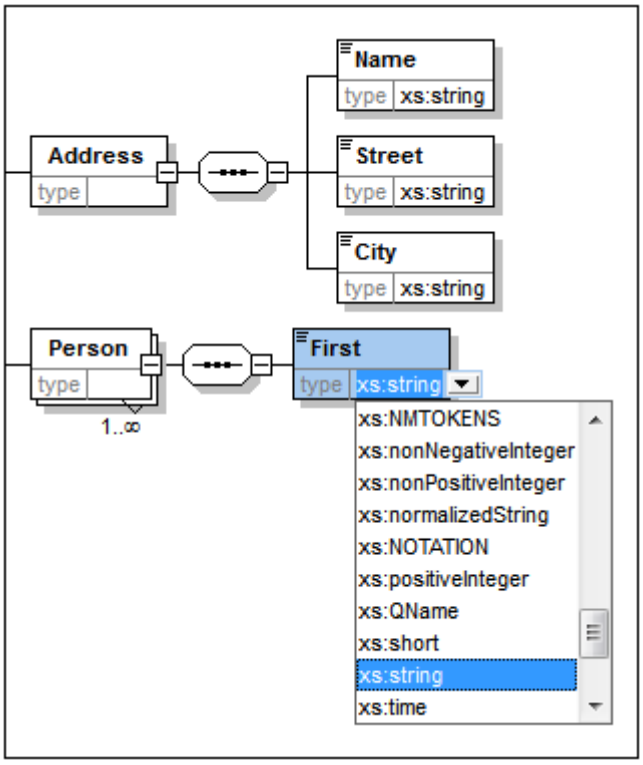
Address 要素のコンテンツが定義されています。以下の simpleType 子要素を含む必要のある **Person** 要素のコンテンツを定義する必要があります：**First**、**Last**、**Title**、**PhoneExt**、および **Email**。これらの要素は **Title** 以外は必須で指定された順序で発生する必要があります。型 **xs:integer** であり、2 桁の数値に制限されている **PhoneExt** 以外の型 **xs:string** である必要があります。

Person のコンテンツモデルを作成するには以下を行います：

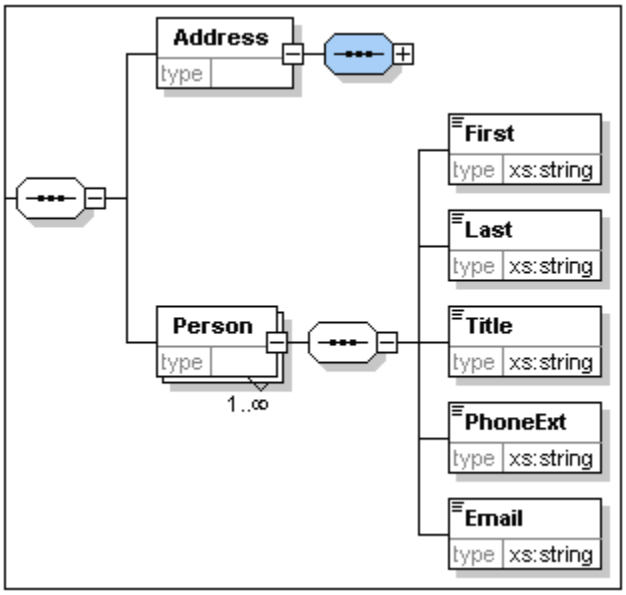
1. **Person** 要素を右クリックし、コンテキストメニューを表示し「**子を追加 | シーケンス**」を選択します。これによりシーケンスコンポジッターが挿入されます。
2. シーケンス要素を右クリックして「**子の追加 | 要素**」を選択します。

 (C) 2015-2021 Altova GmbH

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



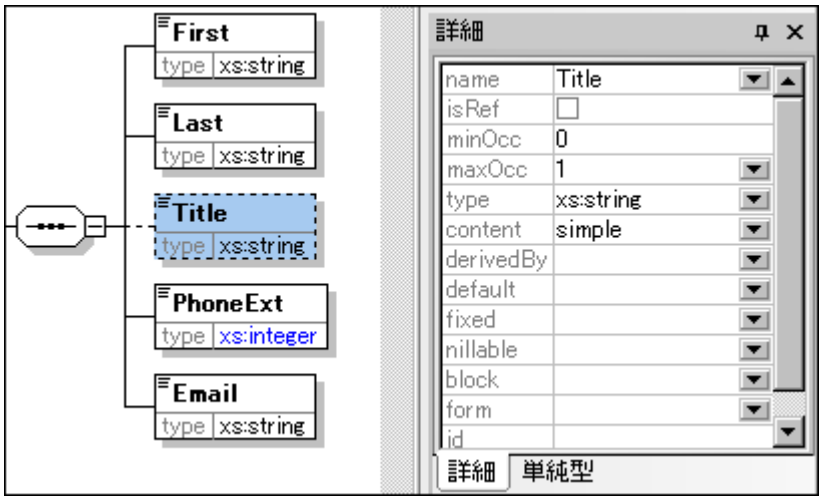
4. **xs:string** エントリをドロップダウンリストから選択するか、*Type* フィールドに入力します。
5. ドラッグアンドドロップにより、その他4つの要素を作成します。それぞれ**Last**、**Title**、**PhoneExt**、および**Email** と名前を付けます。



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

要素を省略可能にする

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

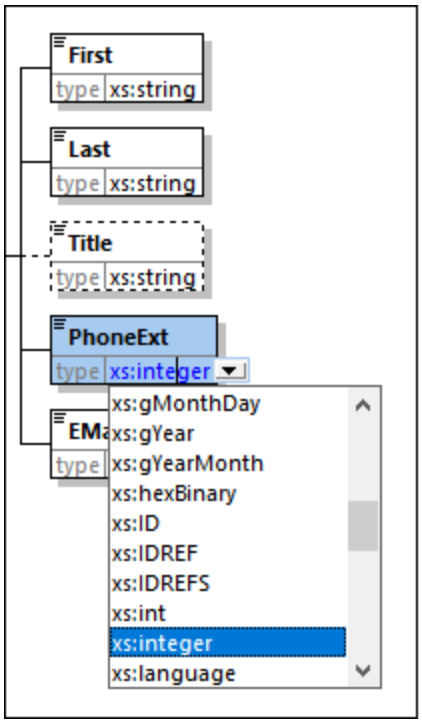


詳細入力ペインにて、**minOcc=0** と**maxOcc=1** と表示され、要素が省略可能であることを示します。コンテキストメニューを使う方法以外にも、**minOcc=0** をセットして要素を省略可能にすることができます。

要素の内容を制限する

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

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



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

以下の点に注意してください

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

2.3 XML スキーマ: 高度な機能

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

目的

このセクションでは、以下のことを学習します:

- スキーマの要素としても使用できる複合型ならびに単純型 56 の扱い方
- グローバル要素 63 を作成し、他の要素からの参照を行う
- 属性 65 とその列挙値を含むプロパティーを作成する

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

このセクションで使用されるコマンド

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

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

2.3.1 複合型と単純型を扱う

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

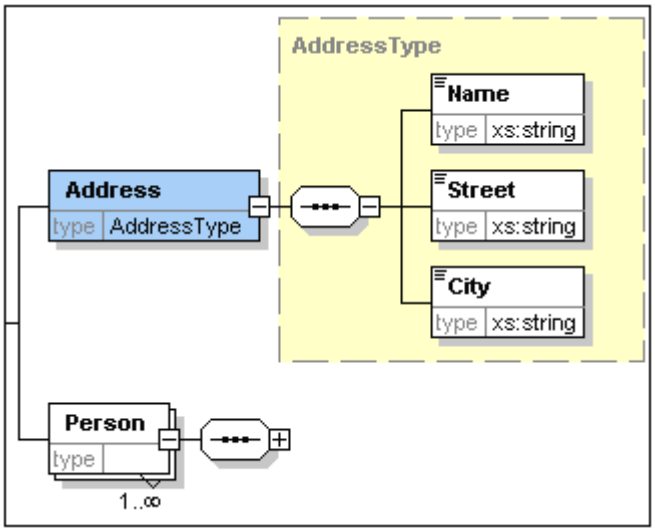
このセクションでは、グローバル複合型と作業します。複合型は単純に他の要素および/または属性を含む要素の型定義です。最初にグローバルレベルで複合型を作成し、コンテンツモデルにインポートし、拡張します。このチュートリアルで後にグローバル要素について学ぶことができます。

グローバル複合型の作成

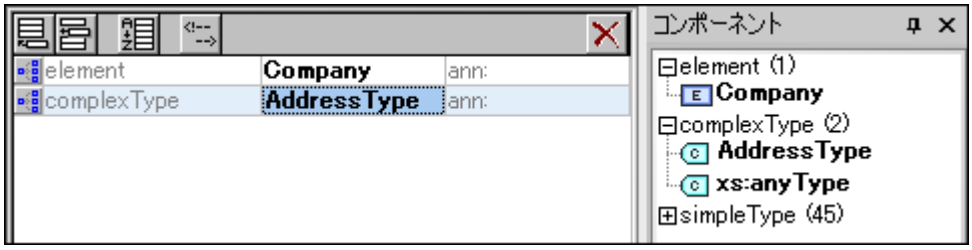
既に定義した(`Name`、`Street`、 `City` 要素を含む) 基本的な `Address` 要素は住所の書式として色々な場所で利用できます。そこで、これら要素の再利用をすることができる複合型を定義することにします。

グローバル複合型を作成するには、以下を行います:

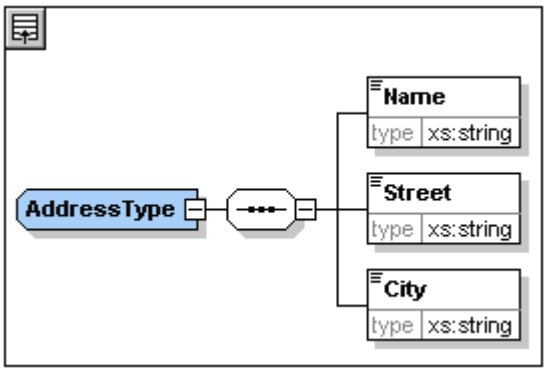
1. コンテンツモデルビューにて **Address** 要素を右クリックする。
2. 現れたコンテキストメニューで「グローバル化 | 複合型」を選択します。**AddressType** という名前のグローバル複合型が作成され、**Company** コンテンツモデルの **Address** 要素がこの型に割り当てられます。**Address** 要素の内容は、黄色のボックスで表示されている AddressType のコンテンツモデルになります。**Address** 要素のデータ型が **AddressType** になっていることに注目してください。



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



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



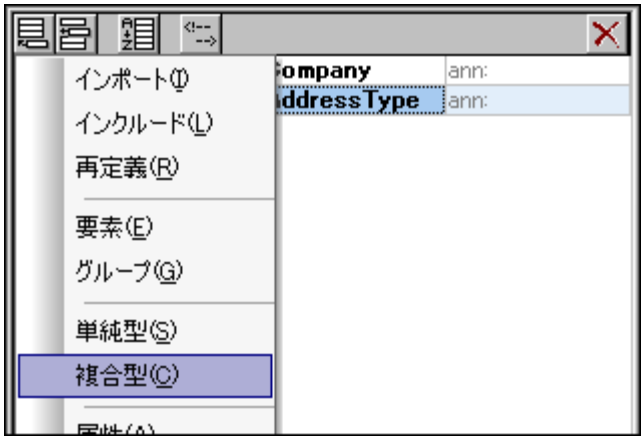
6. 全てのグローバルを表示アイコン をクリックして、スキーマ概要に戻ります。

複合型の定義の拡張

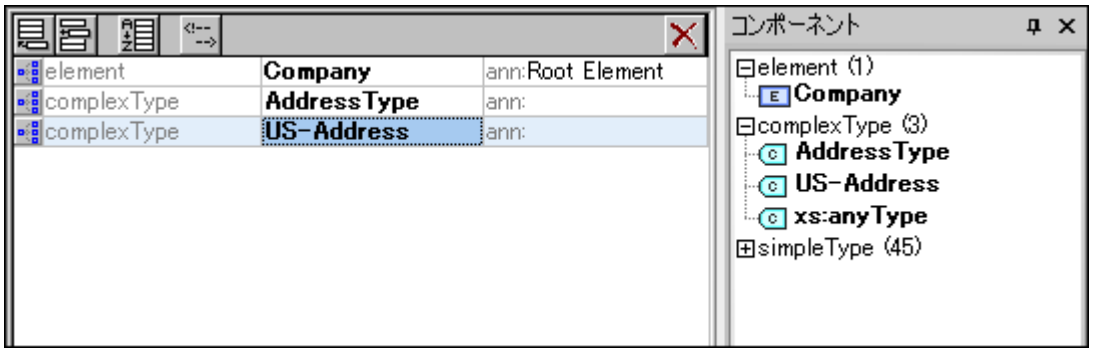
それでは、2つの国に特化した住所を表す為に、グローバル **AddressType** コンポーネントを使用したいと思います。このために、**AddressType** をベースにした新たな複合型を作成し、その定義を拡張します。

以下を行います:

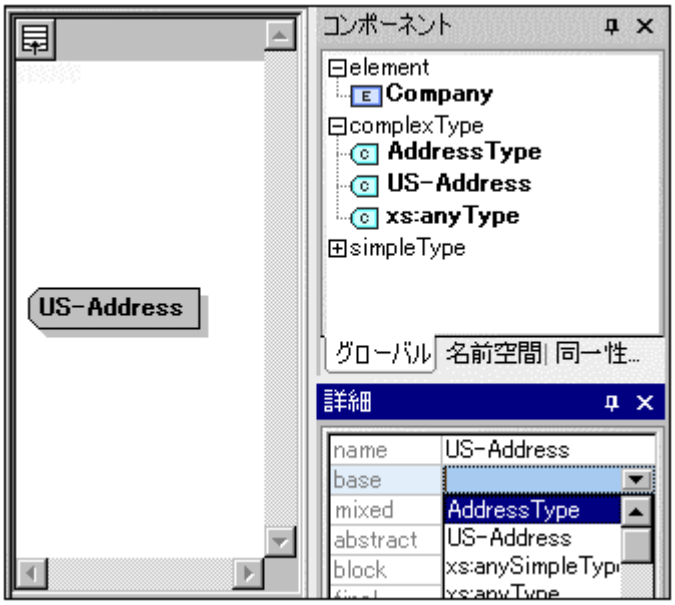
1. スキーマ概要に移動します(コンテンツモデルビューにいる場合、全てのグローバルを表示アイコン をクリックします)。
2. コンポーネントウィンドウの左上にある追加アイコン をクリックします。以下のメニューが開きます:



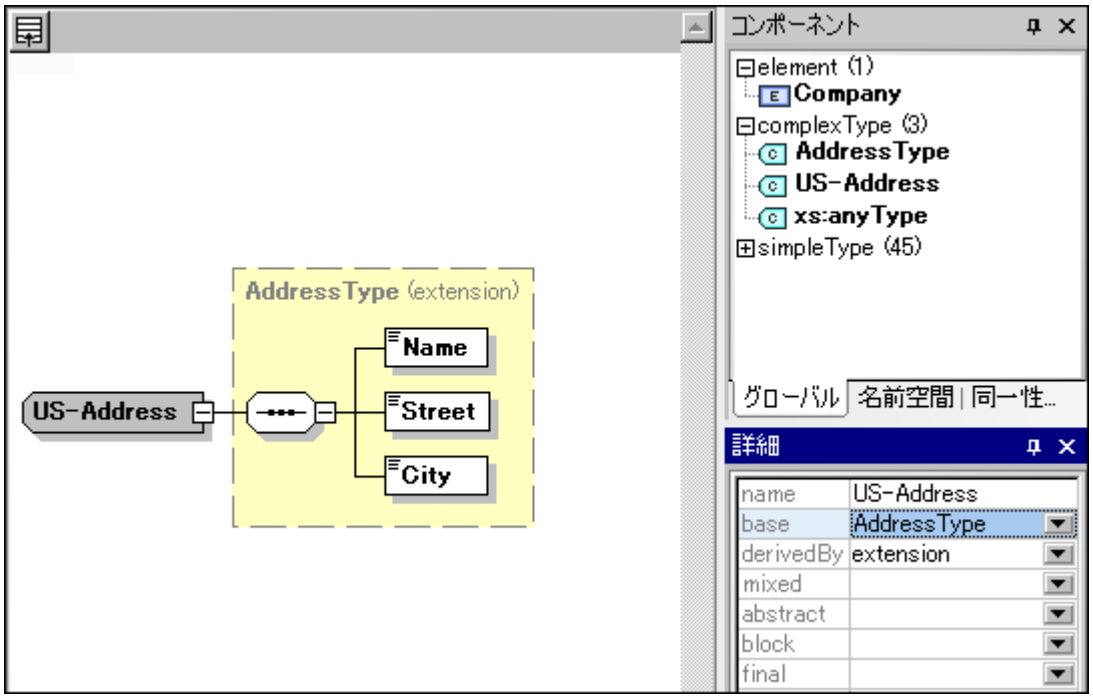
3. メニューから**複合型**を選択します。コンポーネントリストに新しいレコードが現れ、コンポーネントの名前を入力するようにカーソルがセットされます。
4. **US-Address** と入力し、「**Enter**」で確定します。(ハイフン(-)を入力する代わりに空白(スペース)を入力した場合、要素名が赤くなり、不正な文字であることが示されます)



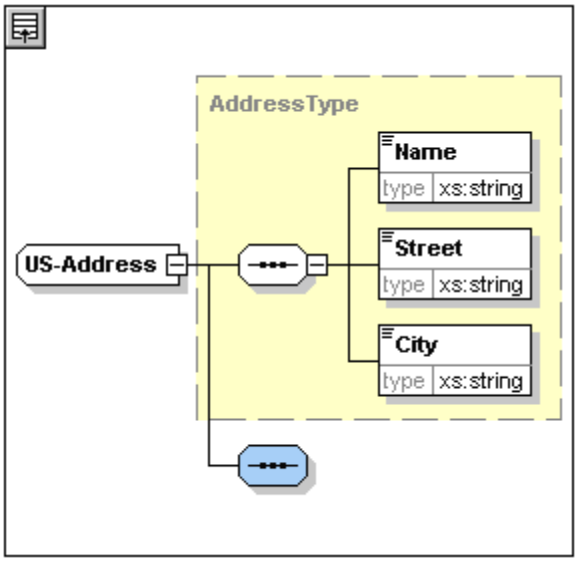
5. **US-Address** のコンテンツモデルビューアイコン をクリックし、新しい複合型のコンテンツモデルを表示します。コンテンツモデルは(以下のスクリーンショットにあるように)空です。
6. 詳細入力ヘルパーにて、base コンボボックスをクリックし、**AddressType** エントリを選択します。



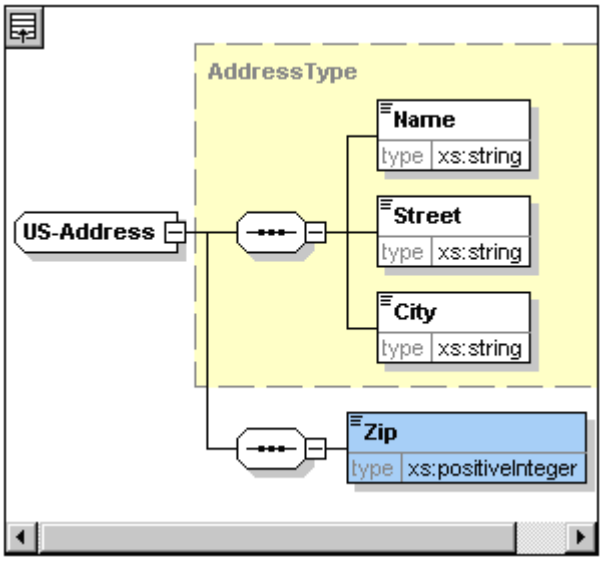
コンテンツモデルビューには **US-Address** のコンテンツモデルとして **AddressType** が表示されます。(*下のスクリーンショット*)。



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



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



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

グローバル単純型

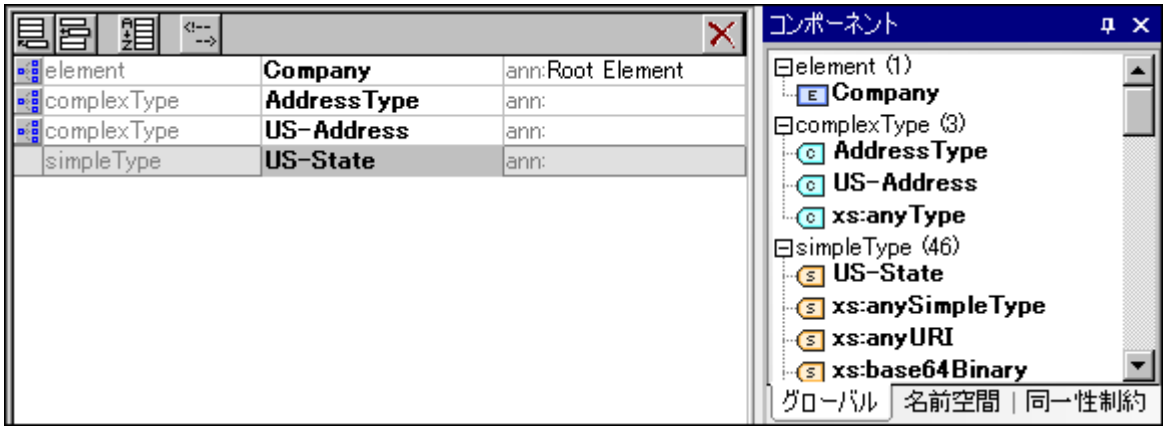
複合型の `US-Address` が複合型の `AddressType` をベースにしたように、単純型をベースにすることもできます。その利点はグローバル複合型の場合と同じで、単純型が再利用されることがあります。単純型を再利用するためには、その単純型をグローバルに定義しなければなりません。このチュートリアルではアメリカの州のコンテンツモデルを単純型として定義することにします。この単純型は他の要素のベースとして使用されることになります。

グローバル単純型の作成

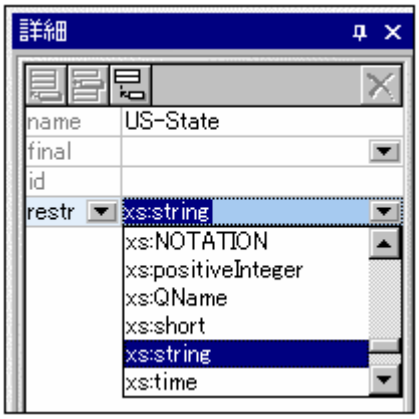
グローバル単純型の作成は、グローバルコンポーネントのリストに新しい単純型を追加し、名前を与え、データ型を定義することから構成されています。

グローバル単純型を作成するには以下をおこないます:

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



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



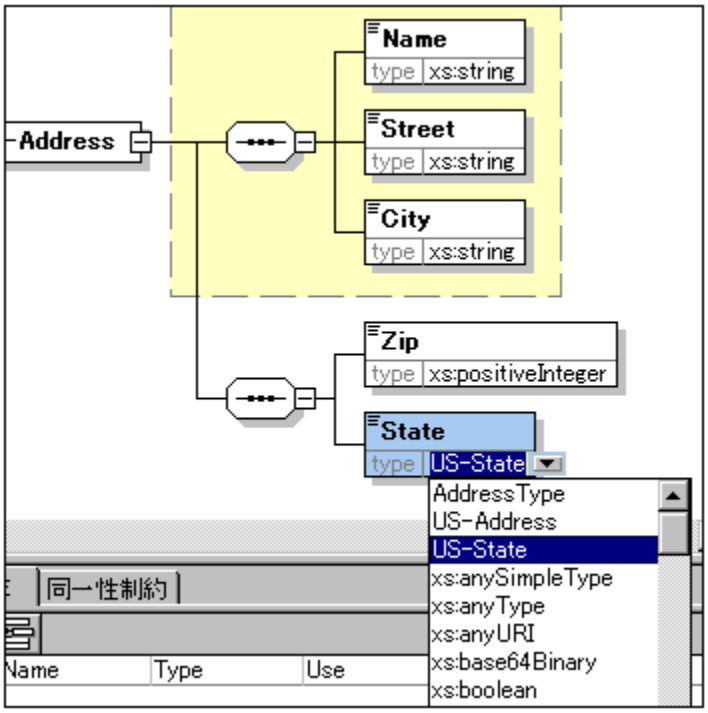
`xs:string` のデータ型である、`US-State` という名前の単純型が作成されます。このグローバルコンポーネントは、`US-Address` のコンテンツモデル内で使用することができます。

コンテンツモデル内でグローバル単純型を使用する

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

次を行います:

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



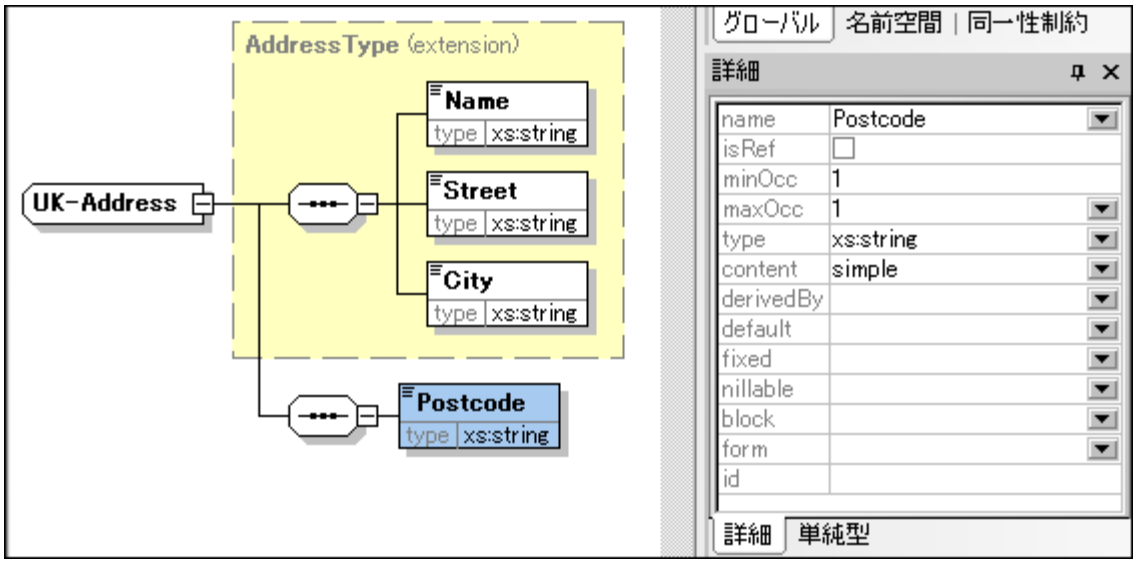
State 要素は、**US-State** 単純型をベースにします。

第 2 の複合型を AddressType をベースにして作成する

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

次を行います:

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

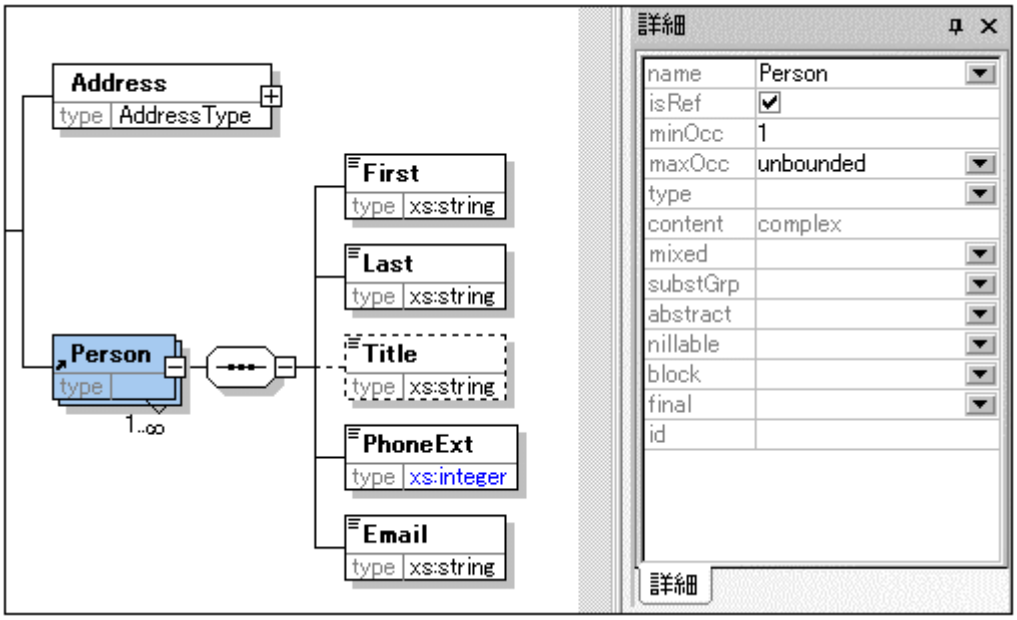


メモ　このセクションでは、コンテンツモデル定義で使用されるグローバル単純型 および複合型を作成しました。グローバル型の利点は、複数の定義で再利用できることです。

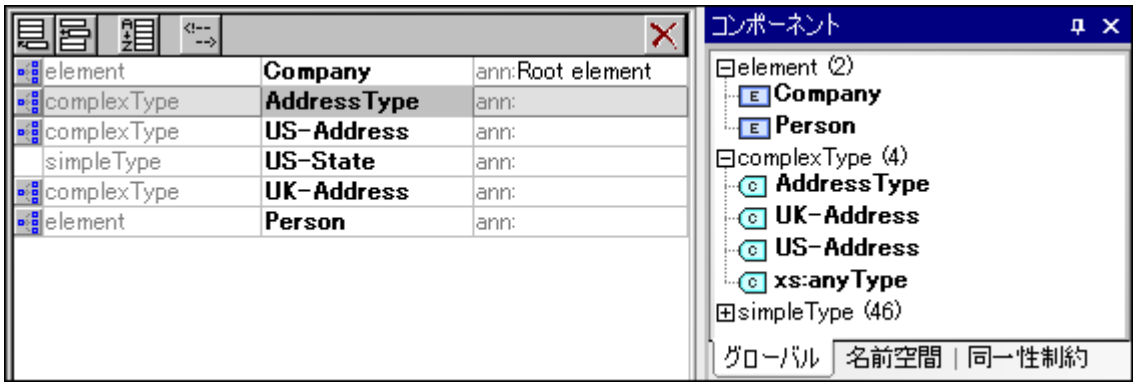
2.3.2　グローバル要素の参照

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

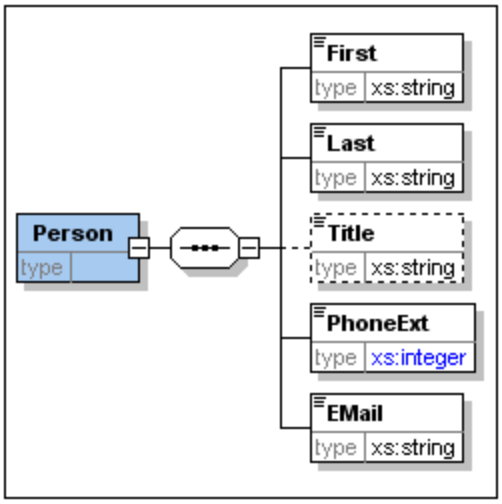
1. 全てのグローバルを表示アイコン をクリックして、スキーマ概要に移動します。
2. **Company** 要素のダイアグラム表示アイコン をクリックします。
3. **Person** 要素を右クリックし、「グローバル化 | 要素」を選択します。要素に小さな矢印アイコンが表示され、この要素がグローバルに宣言された **Person** 要素を参照していることを表します。詳細入力ヘルパーには **isRef** チェックボックスにチェックが入ります。



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



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



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

以下の点に注意してください：

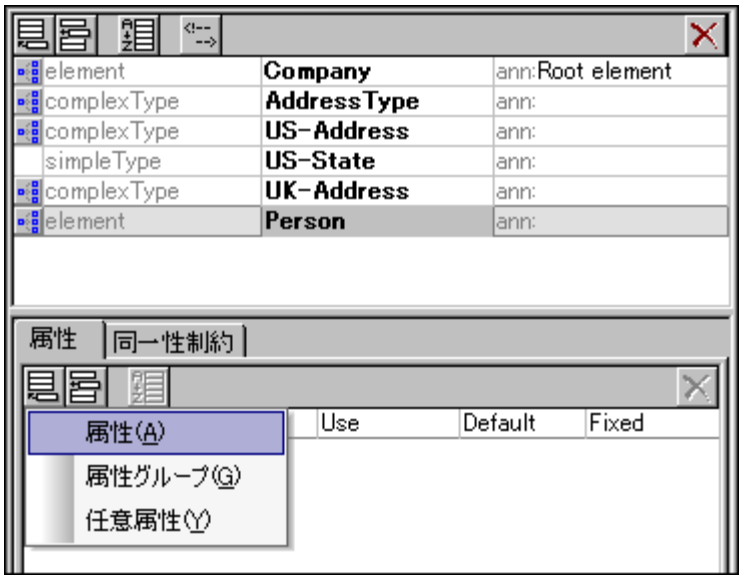
- グローバル要素を参照する要素は、参照するグローバル要素と同じ名前でなければなりません。
- グローバル定義はXMLドキュメントのどこでコンポーネントが使用されるかの記述は行わず、コンテンツモデルの記述だけを行います。グローバル定義が他のコンポーネントから参照された時に、XMLドキュメント内での場所が指定されます。
- グローバルに記述された要素は異なる場所で再利用することが可能です。グローバル複合型の場合とお違い、グローバル要素そのものを変更することなくコンテンツモデルの変更を行うことはできません。グローバル要素を参照する要素のコンテンツモデルを変更した場合、グローバル要素のコンテンツモデ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

2.3.3　属性と列挙型

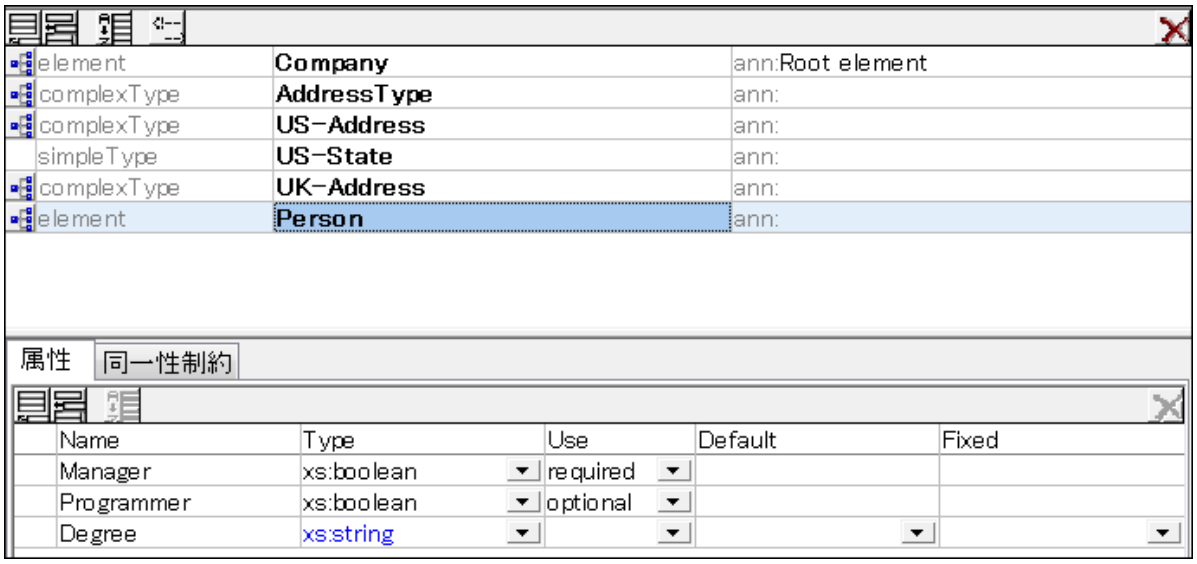
このセクションでは、属性と列挙型の属性の作成方法について学習します

要素属性の定義

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



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



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

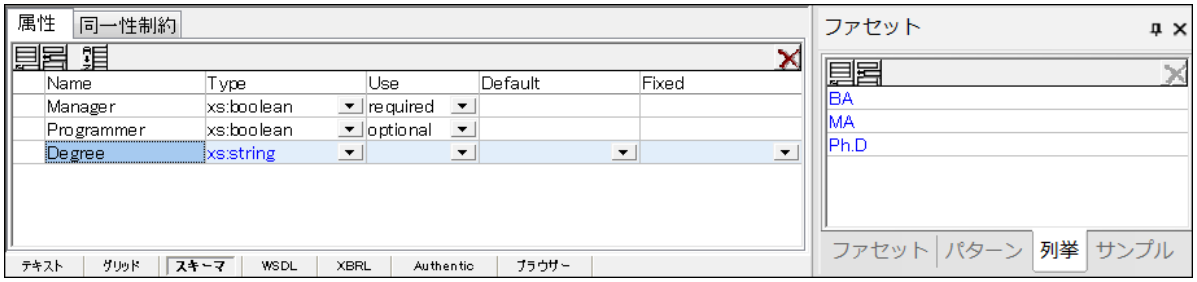
属性の列挙型の定義

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

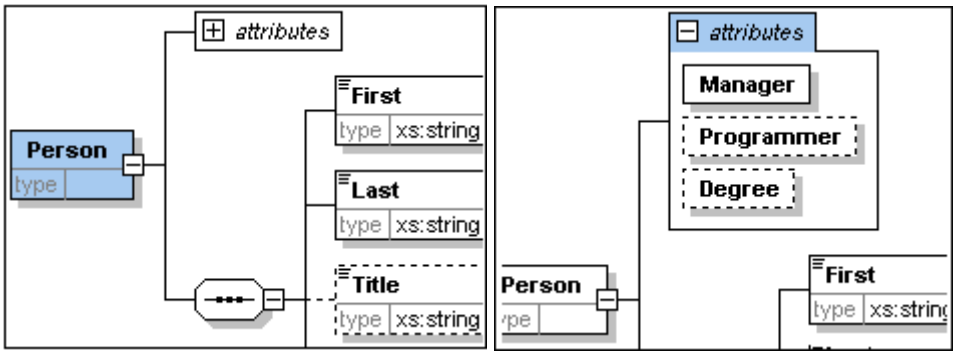
次を行います:

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

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



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



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

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

完成したスキーマの保存

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

2.4 XML スキーマ：XMLSpy の機能

XMLスキーマを完成させた後は、ナビゲーションショートカット 68 とXMLSpy から生成することができるスキーマドキュメンテーションの 70 学習をしましょう。これらの機能についてはこのセクションのサブセクションで説明されています。

このセクションで使用されるコマンド

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

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

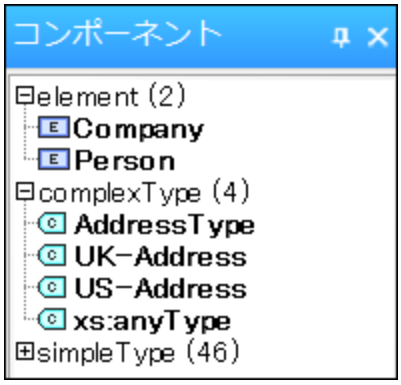
2.4.1 スキーマナビゲーション

このセクションでは効果的にスキーマビューをナビゲートする方法を学習します。

グローバルコンポーネントのコンテンツモデルを表示する

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

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

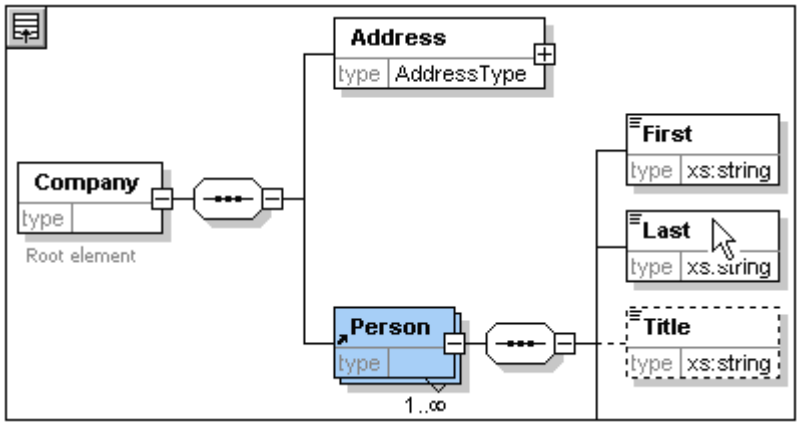


コンポーネント入力ペインにあるその他のグローバルコンポーネント（単純型、属性、属性グループ）をダブルクリックすると、そのコンポーネントはスキーマ概要でハイライトされます（そのようなコンポーネントはコンテンツモデルを持たないため）。

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

参照している要素からグローバル要素に移動する

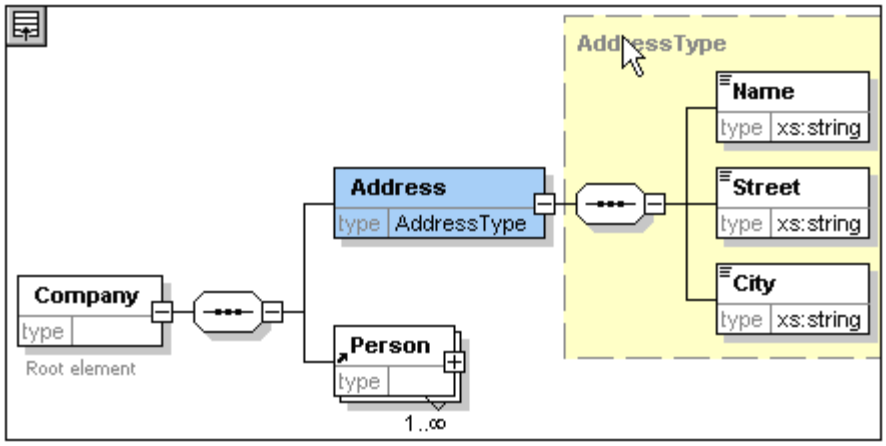
グローバル要素を参照する要素がコンテンツモデルに含まれている場合、「Ctrl」キーを押しながら必要な要素をダブルクリックすることで、そのグローバル要素またはその要素のコンポーネントに移動することができます。



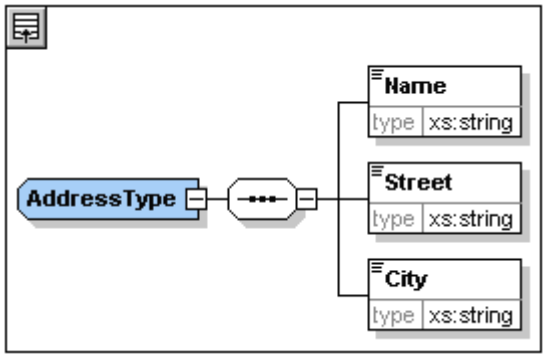
`Last` 要素がハイライトされたとき、関連する情報が入力ヘルパーならびに情報ウィンドウに表示されます。

複合型の定義に移動する

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



この操作で複合型のコンテンツモデルビューに移動することができます。



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

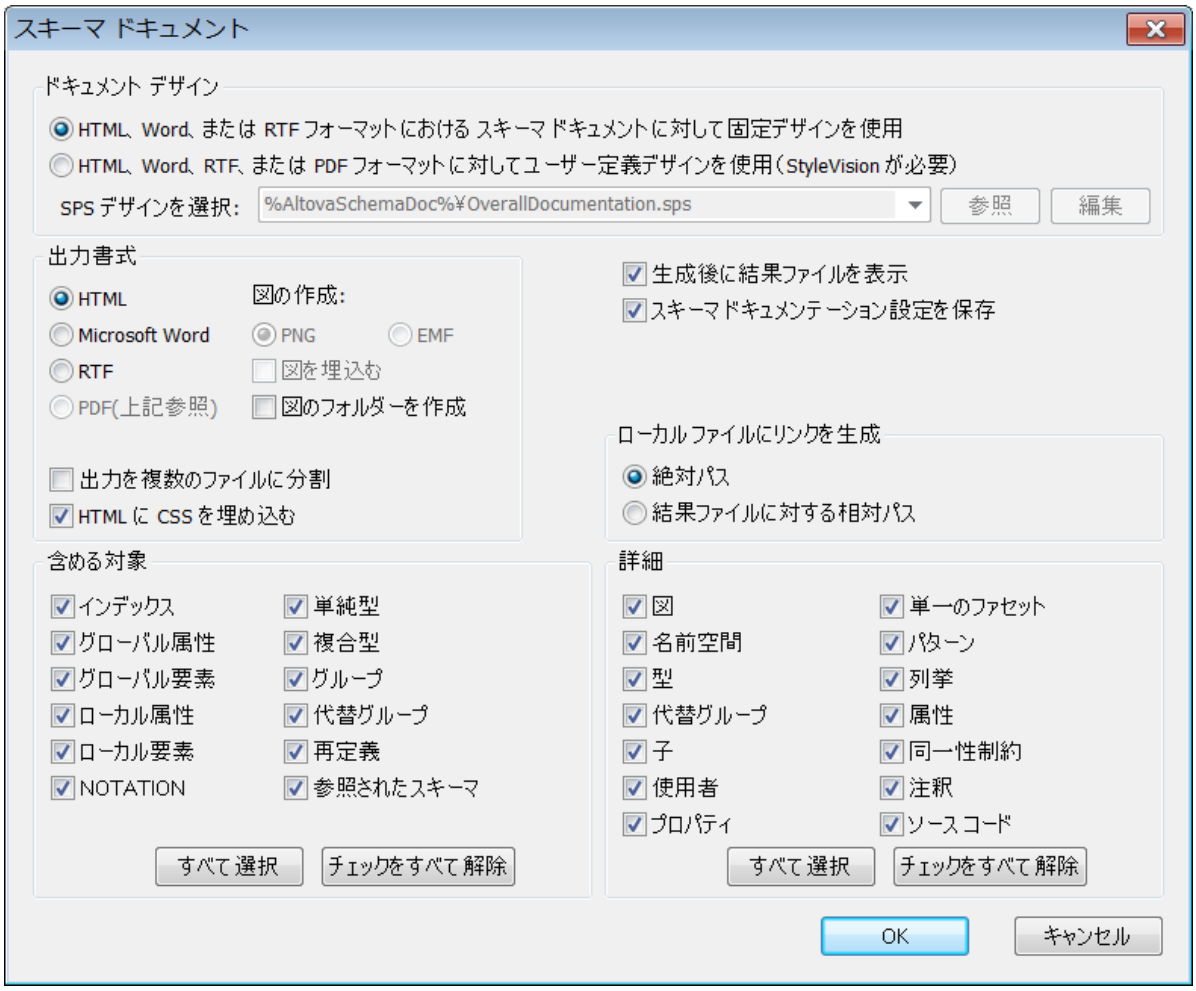
2.4.2 スキーマドキュメンテーション

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

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

以下を行います:

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



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

(C) 2015-2021 Altova GmbH

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

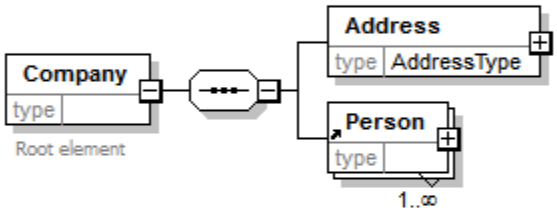
Schema **AddressLast.xsd**

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

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

element Company

diagram

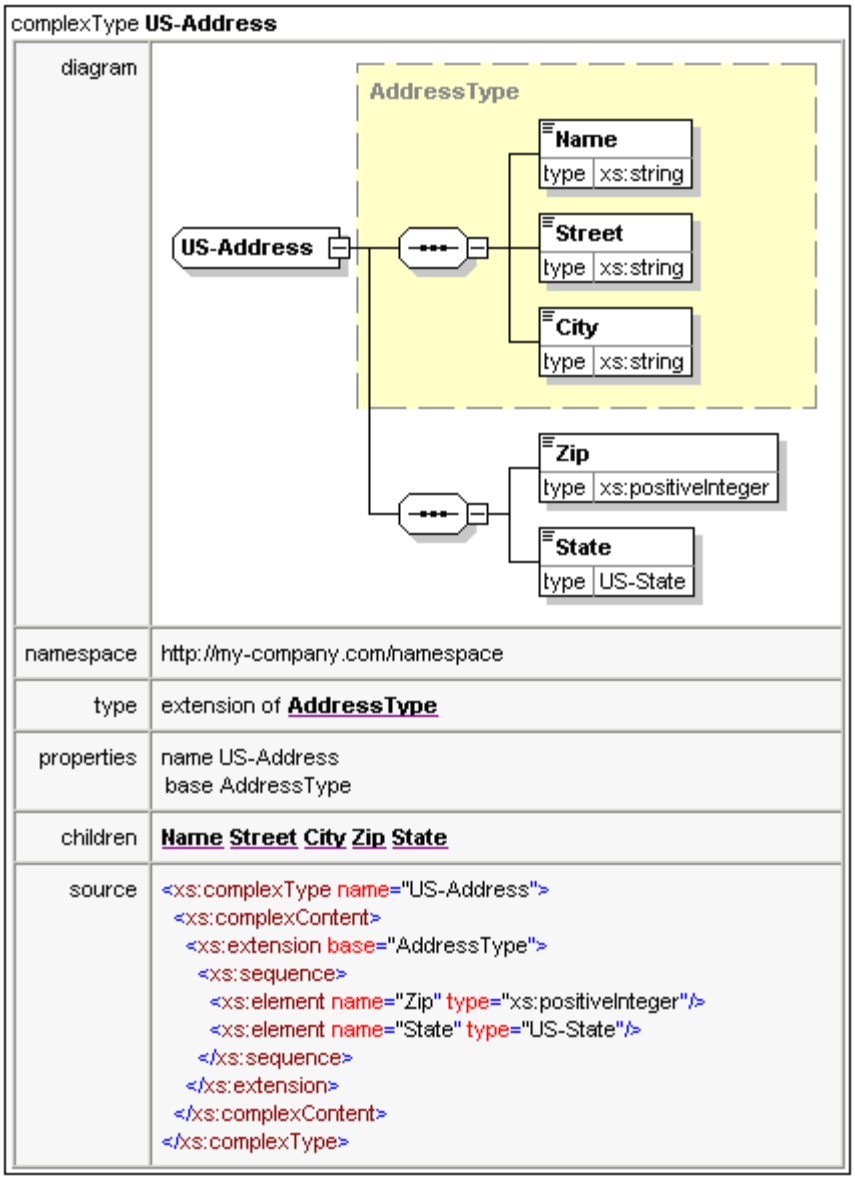


namespace	http://my-company.com/namespace
properties	content complex
children	**Address** **Person**
annotation	documentation Root element

source

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

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



上の図は複合型がどのように表示されるか示しています。

element **US-Address/Zip**

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

element **US-Address/State**

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

simpleType **US-State**

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

上の図は要素や単純型がどのように表示されるか示しています。

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

2.5 XML ドキュメント

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

目的

このセクションでは以下のことを学習します:

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

このセクションで使用されるコマンド

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

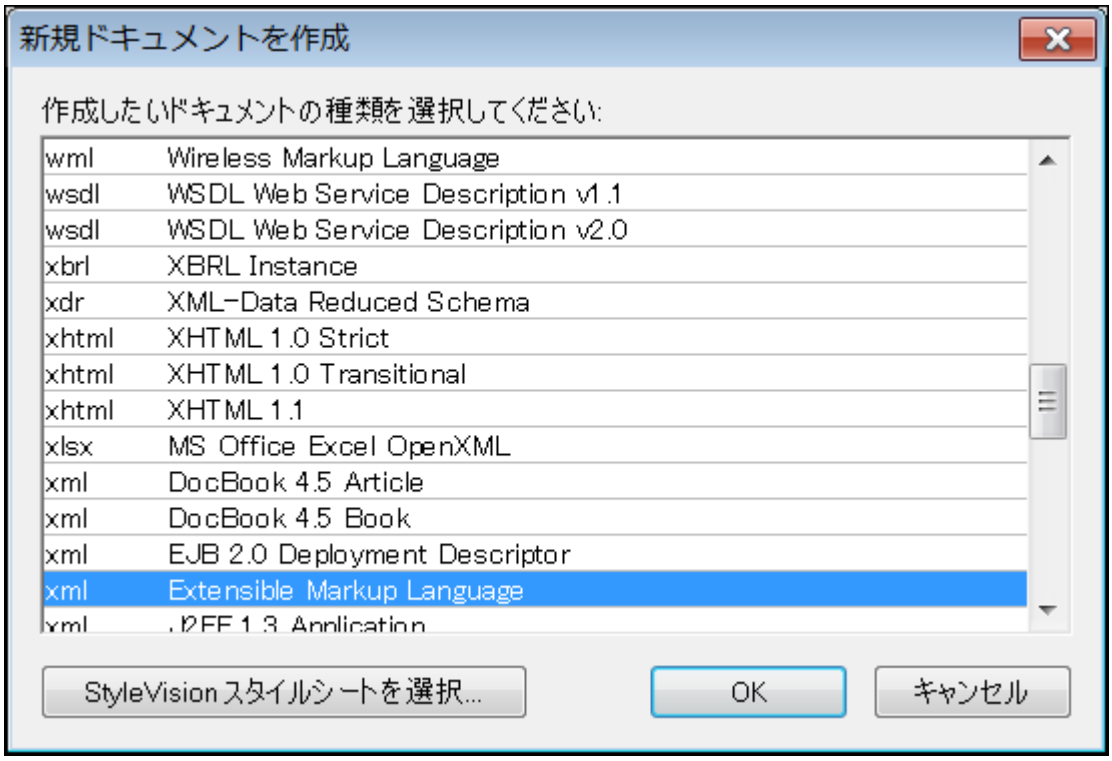
	**ファイル	新規作成** 新しい種類のXMLファイルを作成します。
	**表示	テキストビュー** テキストビューに切り替えます。
	**表示	グリッドビュー** 拡張グリッドビューに切り替えます。
	**XML	テーブル	テーブルとして表示** 1つの要素型の複数の出現をフラットなテーブルで表示します。このビューはデータベース/テーブルビュー(または単純にテーブルビュー)といいます。アイコンはグリッドビューで一般的なビューとテーブルビューを切り替えるために使用されます。
	F7。整形式のチェックを行います。
	F8。割り当てられたDTD またはスキーマに対してXMLドキュメントの検証を行い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

2.5.1 新規 XML ファイルの作成

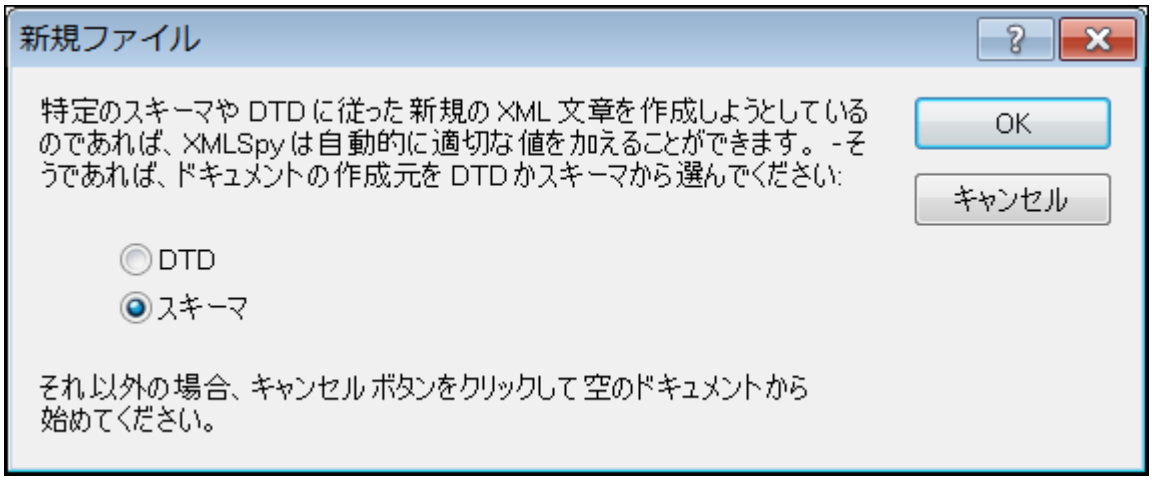
XMLSpy にて新しいXML ファイルを作成する際、スキーマ(DTD またはXML スキーマ)をベースにしたものにするかどうか決定することができます。このセクションでは、既に作成した`AddressLast.xsd` スキーマをベースにした新しいファイルを作成します。

新しいXML ファイルを作成する方法:

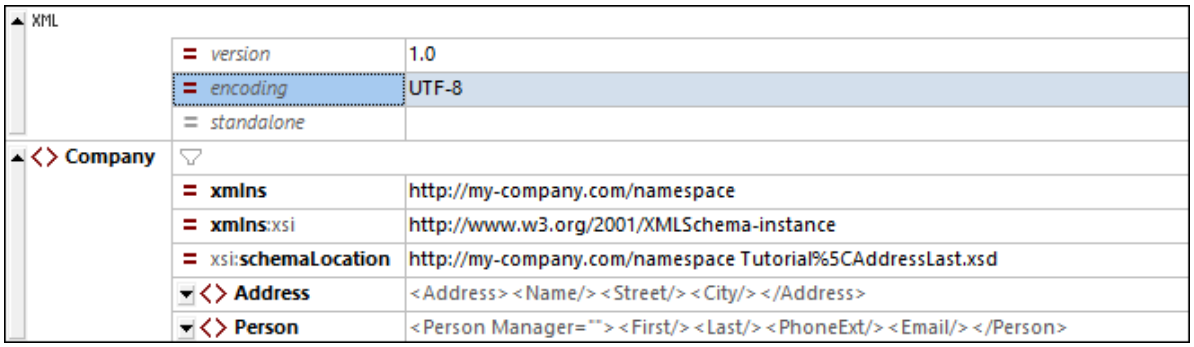
1. メニューオプション[**ファイル | 新規作成**]を選択します。*新規ドキュメント*を作成ダイアログが開かれます。



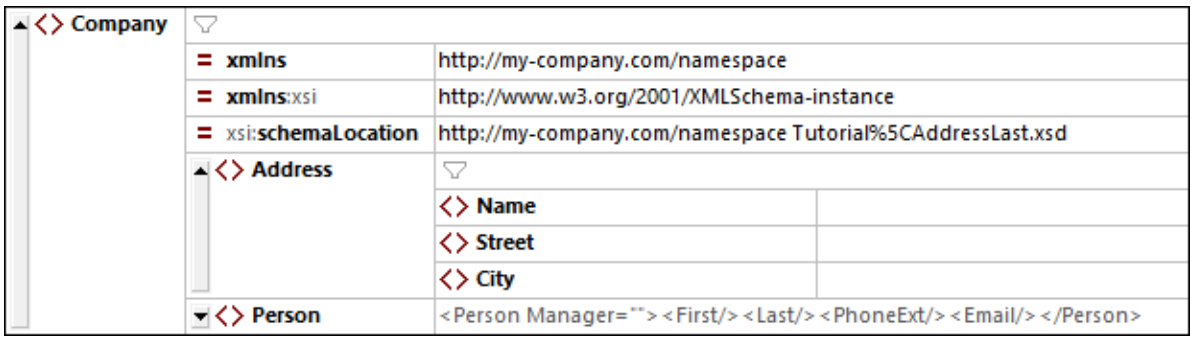
2. ダイアログから *Extensible Markup 言語* エントリ(ジェネリックな XML ドキュメントエントリ) を選択し、「OK」で確定します。ポップアップが表示され、XML ドキュメントのベースにDTD または スキーマを使用するか尋ねられます。



3. スキーマラジオボタンをクリックし、「OK」で確定します。次のダイアログが表示され、XML ドキュメントのベースとなるスキーマファイルを選ぶように促されます。
4. 参照もしくはウィンドウボタンを使用して、スキーマファイルを見つけます。ウィンドウボタンを使うことで XMLSpy にある全てのファイルとプロジェクトを閲覧することができます。`AddressLast.xsd` を選択し(チュートリアルのイントロダクション[29]を参照ください)、「OK」により確定します。スキーマで定義された要素を含む XML ドキュメントがメインウィンドウで開かれます。
5. グリッドタブをクリックしてグリッドビューを選択します。
6. グリッドビューに表示されるドキュメントの構造に注目してください。要素をクリックして選択範囲を狭めてください。ドキュメントは以下のように表示されます:



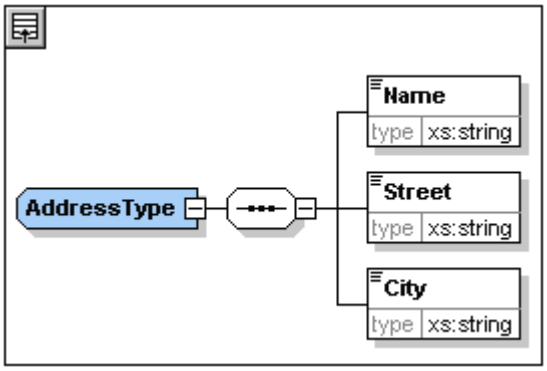
7. **Address** の横にある アイコンをクリックして、**Address** の子要素を表示します。ドキュメントは以下のようになるはずです。ドキュメントは以下のように表示されます:



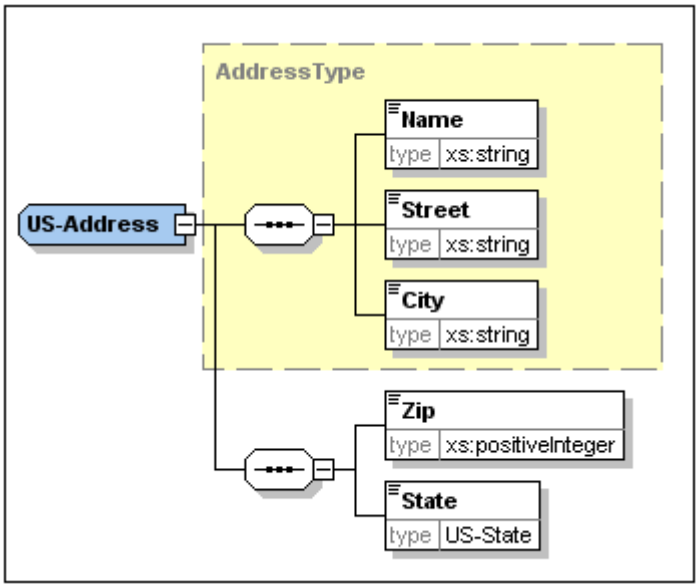
8. メニューオプション [ファイル | 保存] を選択して **Tutorial** フォルダー内に保存します。XML ドキュメントに適切な名前を与えます (例えば、**CompanyFirst.xml**)。**Tutorial** フォルダー内にも既に **CompanyFirst.xml** ファイルがあることご注意してください。作成したファイルを保存する前に、そちらのファイルの名前を変更する必要があるかも知れません。

2.5.2 要素の型の指定

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



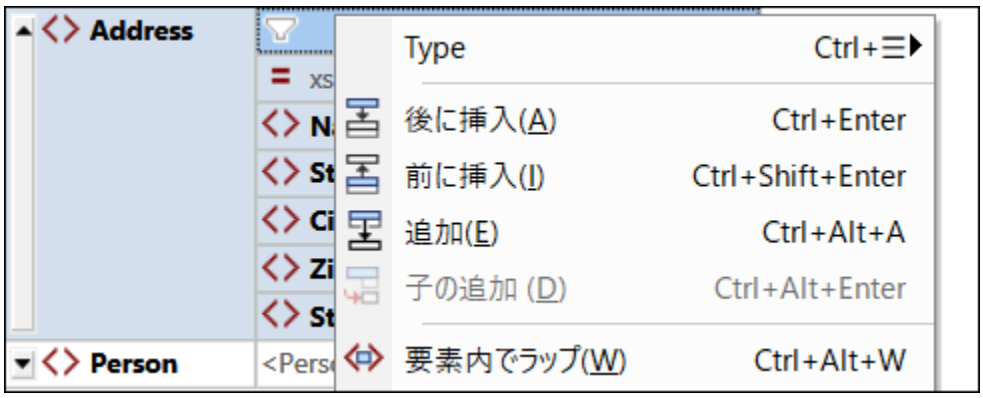
しかしここでは一般的な住所の型より、US または UK の住所を使用します。`AddressLast.xsd` スキーマで `US-Address` と `UK-Address` というグローバル複合型を作成したのを覚えているでしょうか? `US-Address` のコンテンツモデルを以下に示します。



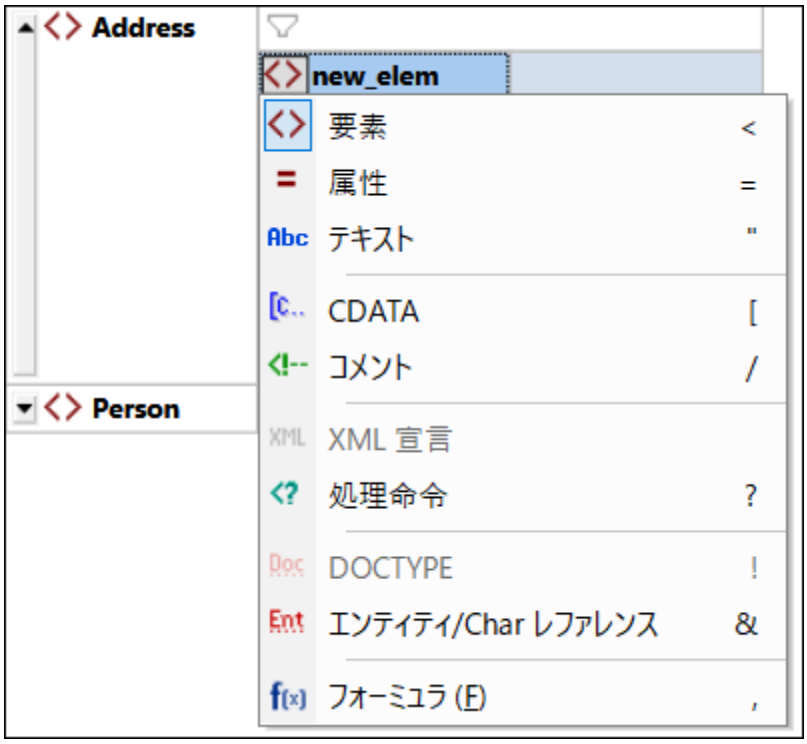
XML ドキュメントにて `Address` 要素が（一般的な AddressType 型ではなく）拡張された Address 型（`US-Address` か `UK-Address`）の要件を満たすためには、必要な複合型を `Address` 要素の属性として指定しなければなりません。

`Address` 要素のこの属性を次のように追加します:follows:

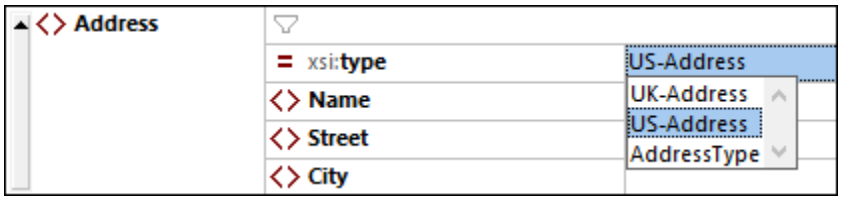
1. XML ドキュメント内で `Name` 要素を右クリックして［**前に挿入**］をコンテキストメニューから選択します（*下のスクリーンショット参照*）。



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



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



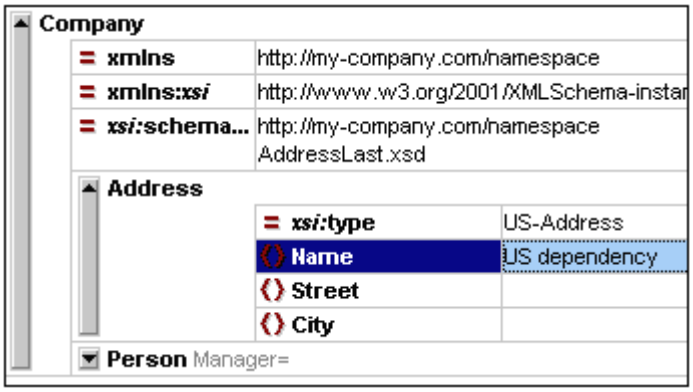
5. `US-Address` を `xsi:type` 属性の値として選択します。

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

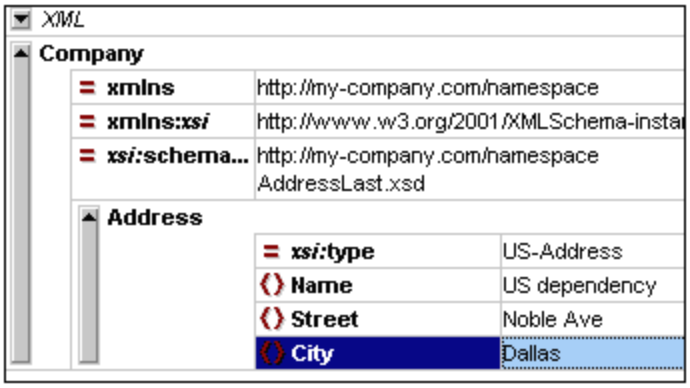
2.5.3 グリッドビューを使ったデータの入力

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

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



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



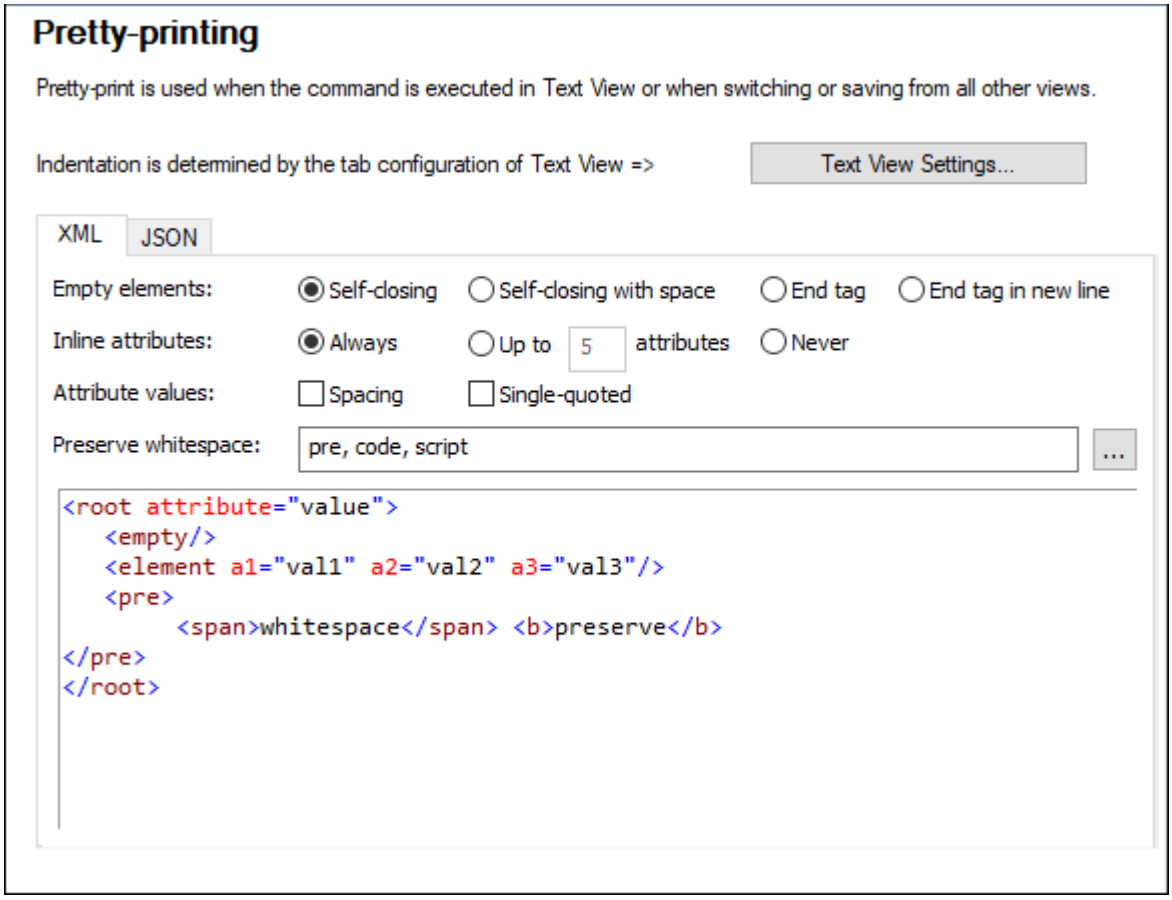
2.5.4　テキストビューを使ったデータの入力

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

ドキュメントのレイアウト

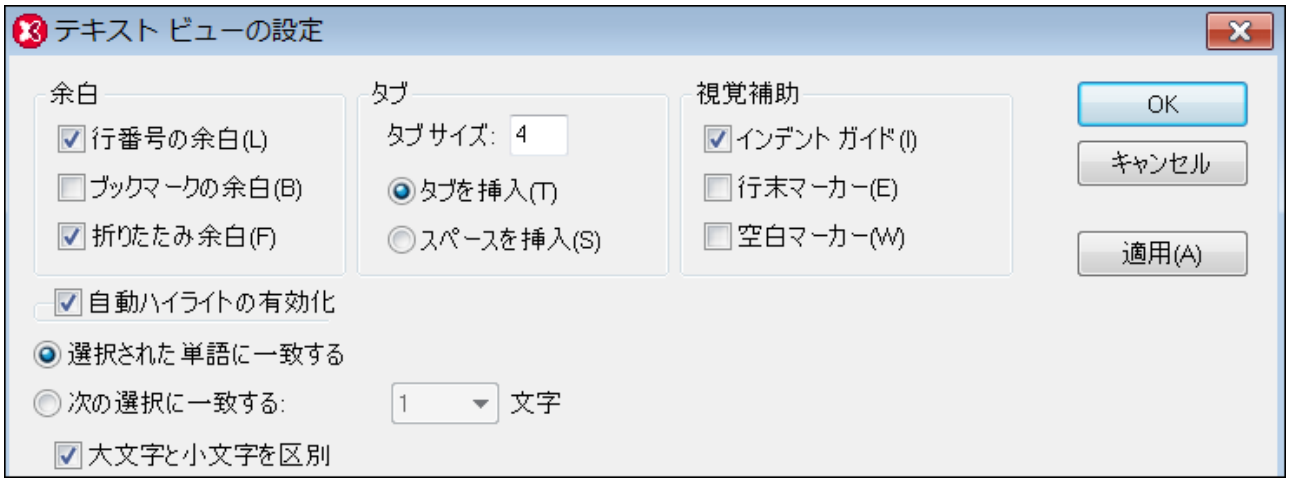
テキストビューのドキュメントのレイアウトは2つの場所で定義されています:

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



テキストビューの設定

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



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

```
1   <?xml version="1.0" encoding="UTF-8"?>
2   <!-- edited with XMLSpy 2021 -->
3   <Company xmlns="http://my-company.com/namespace"
4   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
5   xsi:schemaLocation="http://my-company.com/namespace AddressLast.xsd">
6      <Address xsi:type="US-Address">
7         <Name>US dependency</Name>
8         <Street>Noble Ave.</Street>
9         <City>Dallas</City>
10     </Address>
11  </Company>
```

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

メモ　テキストビューに関連した印刷およびブックマークの機能はこのチュートリアルのテキストビューの設定 36 のセクションで説明されています。

テキストビューでの編集

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

メモ　*編集中の検証* 機能はデフォルトでオンに切り替えられるため、編集中に生成された検証エラーはメッセージウィンドウにエラーメッセージとしてすぐにフラグされます。これらのエラーとメッセージを今の時点では無視してください。バックグラウンドでの検証を必要としない場合、*編集中の検証* をオプションダイアログの検証設定 1015 内でオフに切り替えてください。これを行う場合、ドキュメントをいつでも検証できることに注意してください (このチュートリアルの次のセクション 83 で説明されています)。

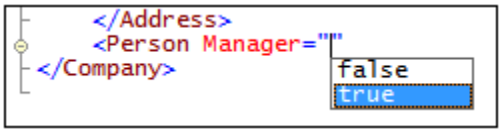
以下を行います:

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

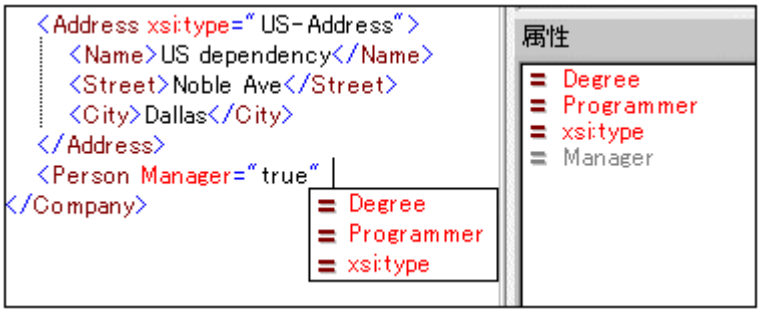
```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSpy 2021 -->
<Company xmlns="http://my-company.com/namespace" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://my-company.com/namespace AddressLast.xsd">
   <Address xsi:type="US-Address">
      <Name>US dependency</Name>
      <Street>Noble Ave.</Street>
      <City>Dallas</City>
   </Address>
   <
     Person

</Company>
```

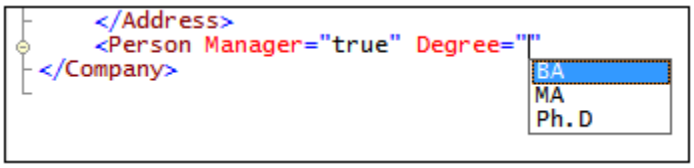
4. **Person** エントリを選択します。**Person** 要素が **Manager** 属性と共に挿入され、**Manager** 属性の値フィールドの内部にカーソルが移動します。
5. **Manager** 属性のポップアップから **true** を選択します。



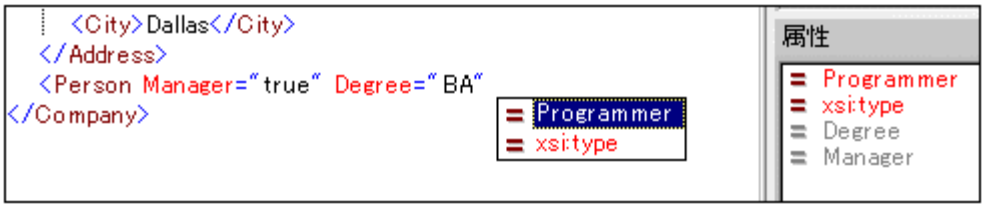
6. カーソルを行末に移動し（例えば「End」キーを使うことができます）、スペースバーを押します。新しいポップアップが開かれ、この位置で許されている属性のリストが表示されます。同様に、属性入力ヘルパーにて使用することのできる属性が赤色で表示されます。**Manager** 属性はグレーになっていますが、それはその属性が既に使われたためです。



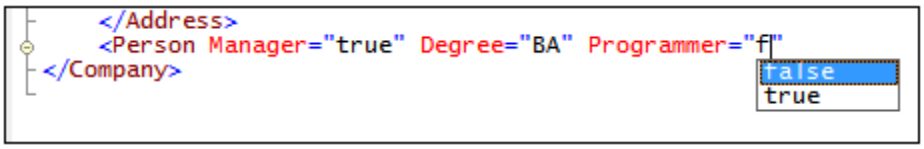
7. **[Degree]** を下向きの矢印を使用して選択し **[Enter]**を押します。これでまた別のポップアップが表示され、既に定義された列挙（**BA**、**MA**、または **PhD**）から選択を行うことができます。（列挙はXML スキーマにより許可されている値です。）



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



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



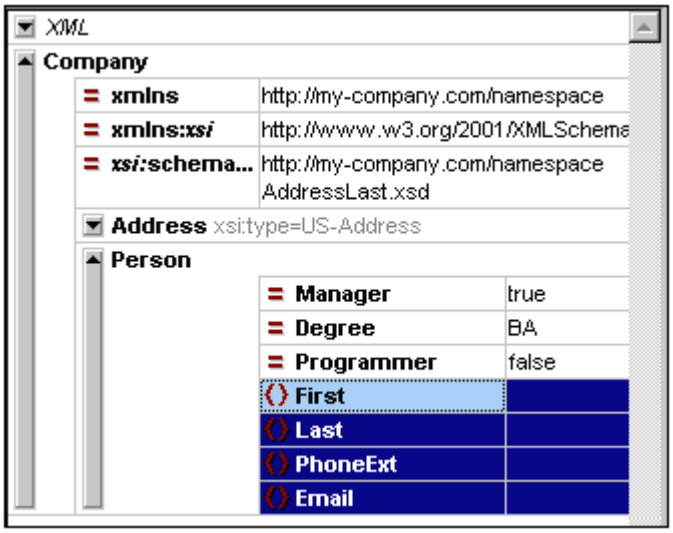
10. 文字 **"f"** を入力し、**[Enter]**を押します。
11. (**End** キーを使用して) カーソルを行末まで移動し、大なり記号 > を入力します。XMLSpy が**Person** に必要な子要素を自動的に挿入します（省略可能な **Title** 要素が挿入されていないことに注目してください）。全ての要素には開始と終了タグがありますが、コンテンツはありません。

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

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

グリッドビューへ切り替え

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



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

2.5.5　ドキュメントの検証

XMLSpy ではXML ドキュメントを 2 つの方法でチェックできます:

- 整形式のチェック
- 検証のチェック

	整形式のチェック

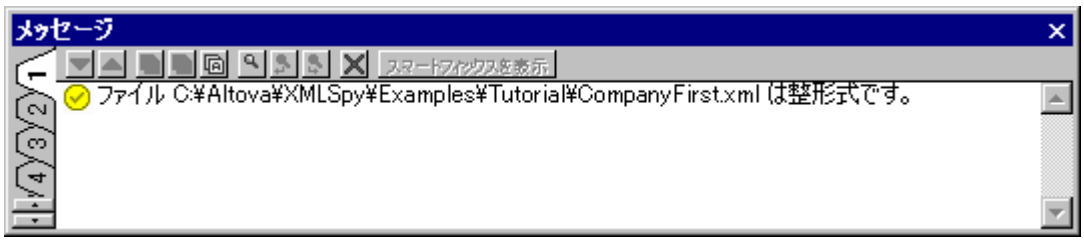
	XML 検証

編集中の検証 機能はデフォルトでオンに切り替えられるため、編集中に生成された検証エラーはメッセージウィンドウにエラーメッセージとしてすぐにフラグされます。バックグラウンドでの検証を必要としない場合、*編集中の検証* をオプションダイアログの検証設定 1015 内でオフに切り替えてください。これを行う場合 XML メニュー内で対応するコマンドを呼び出して整形式チェックと検証チェックを個別にいつでも行えることに注意してください。チュートリアルの一部でこれらのチェックを行う方法が説明されています。

整形式チェック

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

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



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

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

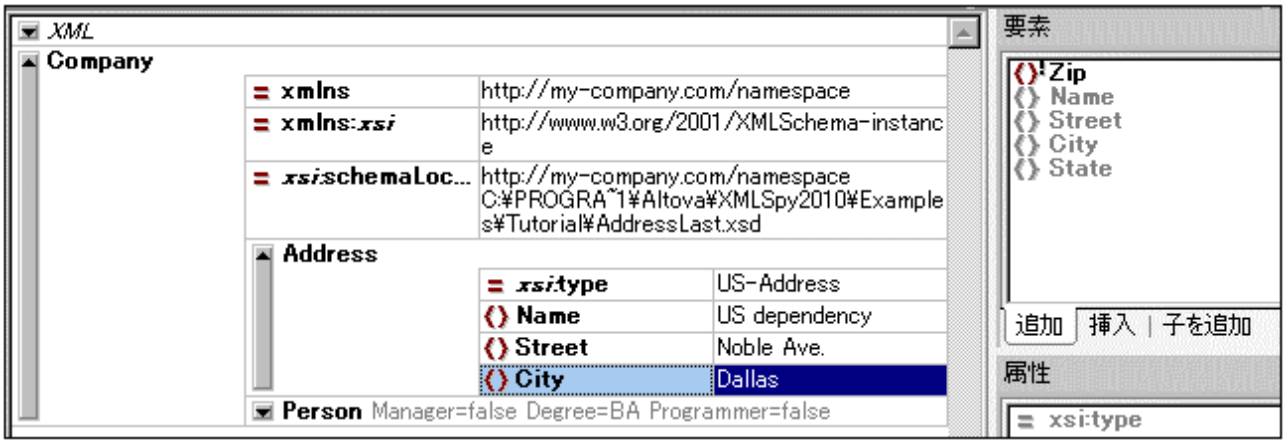
妥当性のチェック

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

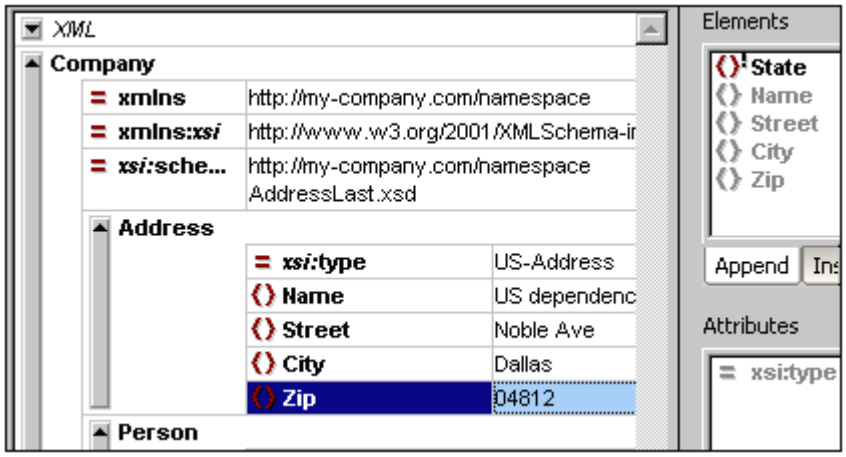
1. グリッドビューを選択します。
2. メニューオプション [XML | **検証**] または **F8** キーをチェックします。（またはツールバー内のコマンドアイコンをクリックすることができます。）メッセージウィンドウにファイルが妥当でないことを示すエラーメッセージが表示されます。`Address` 内の `City` 要素の後に必須要素が期待されます。（`xsi:type` 属性を介してこの `Address` 要素を設定する）スキーマをチェックすると `US-Address` 複合型 は `City` 要素に `Zip` 要素と `State` 要素が続く必要のあるコンテンツモデル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

無効なドキュメントの修正

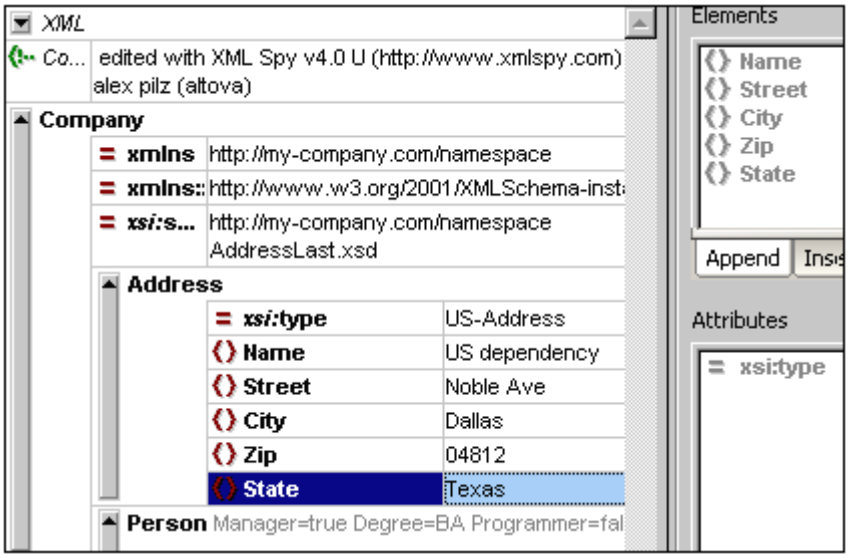
ドキュメントが無効になる個所がスマートフィックス 271 と共に赤色でハイライトされます。この場合の無効な要素は `Address` 要素です。スマートフィックスアイコンを使用するとポップアップが表示されます: *同じコンテンツを持つ不足する要素 <Zip> を追加します。*スキーマをチェックすると `Address/City` 要素が必須要素 `Zip` が後に続くことが必要なことに気が付きます。これをダブルクリックするには `City` 要素を選択して、兄弟入力ヘルパーを確認します。`Zip` 要素が現在のコンテキスト内で要素が必須なことを示す感嘆符と共にプレフィックスされていることに気が付きます。



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



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

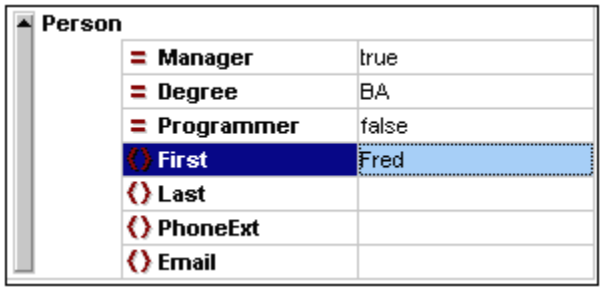


ドキュメントを完成させ、再度検証する

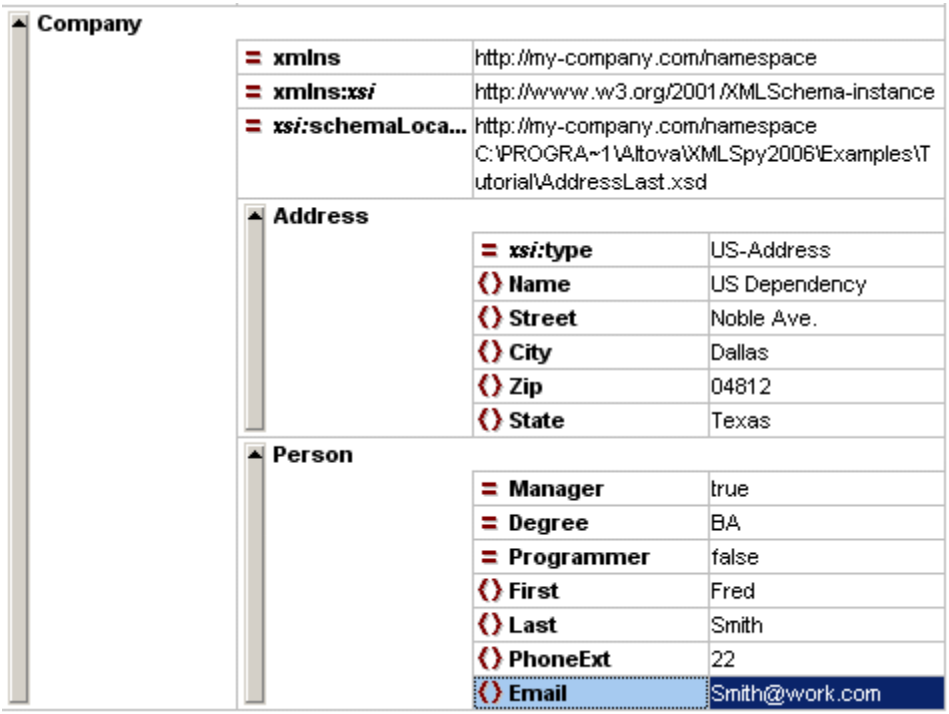
それでは、再度検証を行う前に `Person` 要素の情報を入力し、ドキュメントを完成させましょう。

以下を行います:

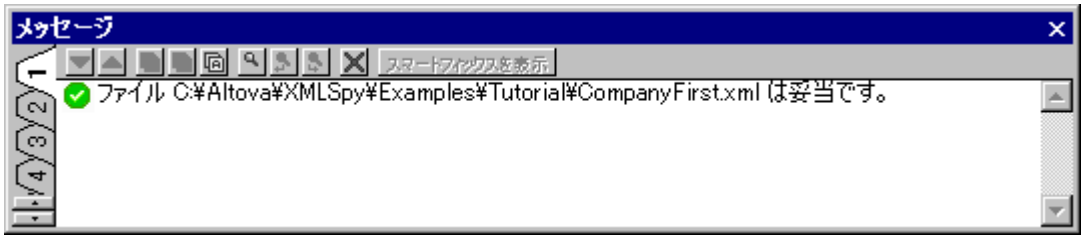
1. **`First`** 要素をクリックして、名前(例えば `Fred`)を入力し、「**`Enter`**」で確定します。



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



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



4. [ファイル | 保存]を使用してファイルを保存します。

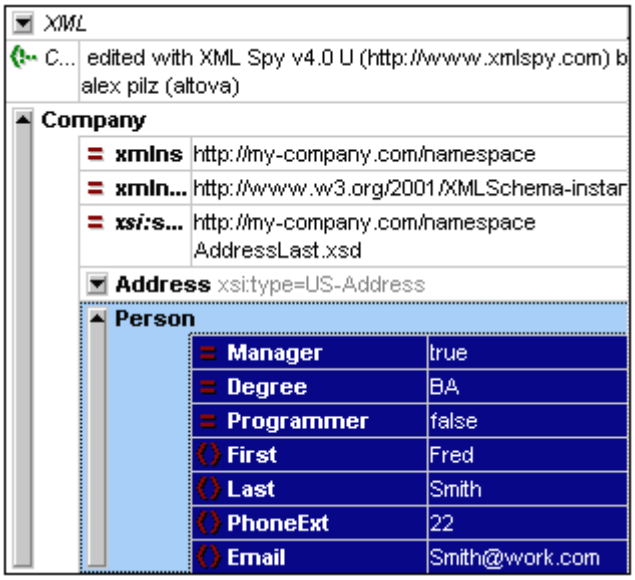
メモ: 保存する際にXMLドキュメントが妥当である必要はありません。不正なドキュメントを保存しようとすると、その旨の警告が表示されます。無効のままの状態で保存したいのであれば、そのまま保存を選択することができます。

2.5.6 要素と属性の追加

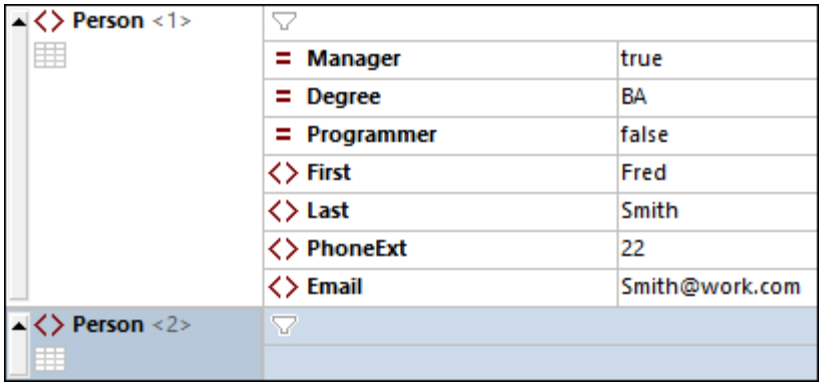
この段階ではドキュメント内に Person 要素は1つしかありません。

新しい Person 要素を追加するには以下を行います:

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



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



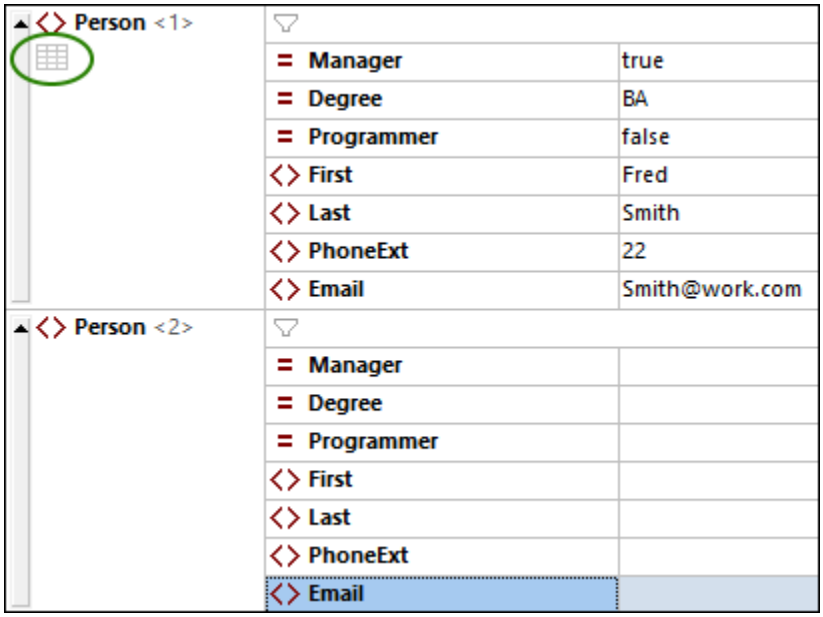
4. `Person` 要素が選択されていると、子入力ヘルパー内で、この要素で使用可能な子属性と要素を確認することができます。属性と要素をダブルクリックして最初の `Person` 要素として同じ子ノードを追加します。`Person` 要素から追加された子要素にメインウィンドウ内のフォーカスが変更されると `Person` 要素の追加の子を以下の2つの方法の1つを使用して追加することができます: (i) `Person` 要素（を選択し）子入力ヘルパーから新規の子を追加してフォーカスを切り替える（ii）追加された子要素のフォーカスを使用して、兄弟入力ヘルパーから兄弟子要素を追加する。両方の入力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

Person <1>		
	Manager	true
	Degree	BA
	Programmer	false
	First	Fred
	Last	Smith
	PhoneExt	22
	Email	Smith@work.com
Person <2>		
	Manager	
	Degree	
	Programmer	
	First	
	Last	
	PhoneExt	
	Email	

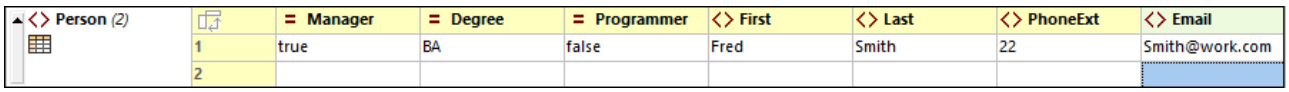
Person 要素の子ノードのために標準のグリッドビューでコンテンツを追加することができますが、Person などの複数の発生がある構造を編集するために適切なグリッドビューのテーブル表示に切り替えることにしましょう。

2.5.7 データベース/テーブルビューでの編集

グリッドビューの中には複数回出現する要素の編集に適したテーブルの表示という特別なビューが存在します。例えば Person 要素には複数の発生があり（*下のスクリーンショット*）これをテーブルとして表示することができます。このような要素をテーブルとして表示するためには要素の最初の発生の **テーブルの表示** アイコンをクリックします。例えば、下のスクリーンショット内では Person 要素の **テーブルの表示** アイコンは緑色の丸枠で表示されています。（または、メニューコマンド XML | **テーブルとして表示** またはグリッドビューツールバー 236 のコマンドのツールバーアイコンをクリックします。）



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



テーブルの表示の利点
テーブルの表示には以下の利点があります:

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

複数の発生を持つ要素をテーブルとして表示する

`Person` 要素をテーブルとして表示するには、以下を行います:

1. `Person` 要素の最初の発生の **テーブルの表示** アイコンを上記の通りクリックします。
2. メニューオプション [XML | **最適な幅**] を選択して、または[グリッドビューツールバー](236) 内の **最適な幅** アイコンをクリックします。

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

テーブルの表示内にコンテンツを入力する方法

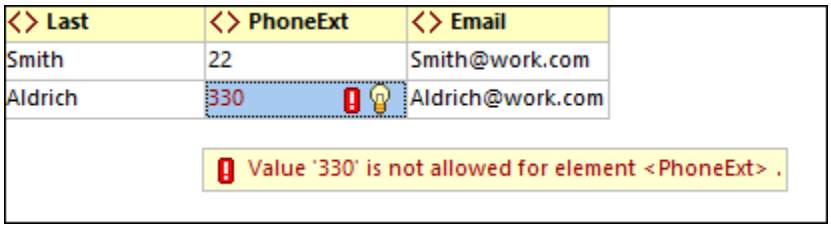
2番目の`Person` 要素にコンテンツを挿入するには、2番目の行内のテーブルセルのそれぞれをダブルクリックして、データを入力します。(例えば、ブール値コンテンツと`Degree` 属性の列挙)などのドロップダウンリストからのオプションを選択できるように、テーブルのセル内でもインテリジェントな機能は有効化されています。

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

動的な検証

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

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

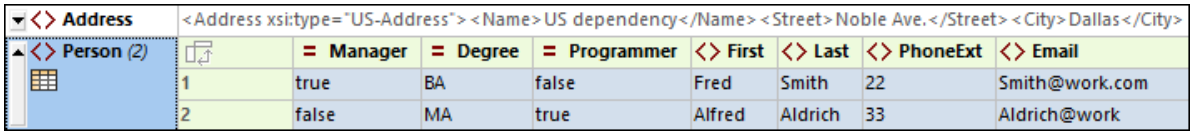


5. スマートフィックスアイコンをクリックするとスマートフィックスオプションが表示されます。無効な値が有効な値と置き換えられ、エラープラグが消えます。

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

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

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



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

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

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

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

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

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

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

列のコンテンツ上のテーブルを並べ替える方法

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

1. **Last** 列をヘッダーをクリックして選択します。

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

2. メニューオプション［XML | 昇順に並べ替え］を選択して、またはグリッドビューツールバー 236内の昇順に並べ替えアイコンをクリックします。列と列を含むテーブルがアルファベット順に並べ替えられます。列はハイライトされたままです。

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

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

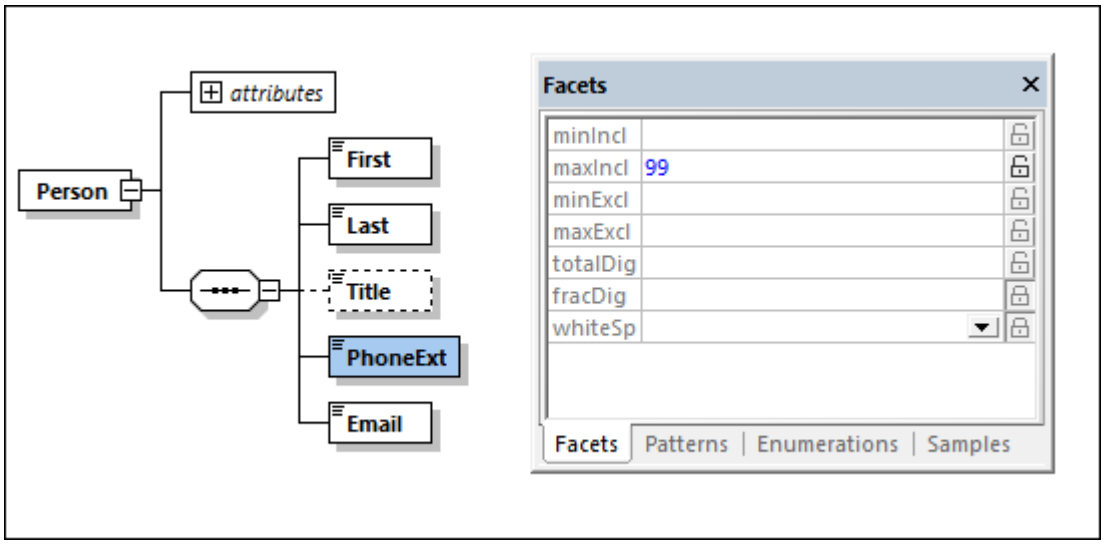
2.5.8 スキーマの修正

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

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

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

2. メニューオプション DTD/スキーマ | 定義に移動 を選択してグリッドビューツールバー 236 内の 定義に移動 アイコンをクリックします。関連付けされたスキーマ、この場合は **AddressLast.xsd** が開かれ **PhoneExt** 定義がハイライトされます (*下のスクリーンショット*)。



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

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

2.6 XSLT 変換

目的

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

方法

変換を行うために必要な方法は以下の通りです:

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

このセクションで使用されるコマンド

以下の XMLSpy コマンドがこのセクションでは使用されます:

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

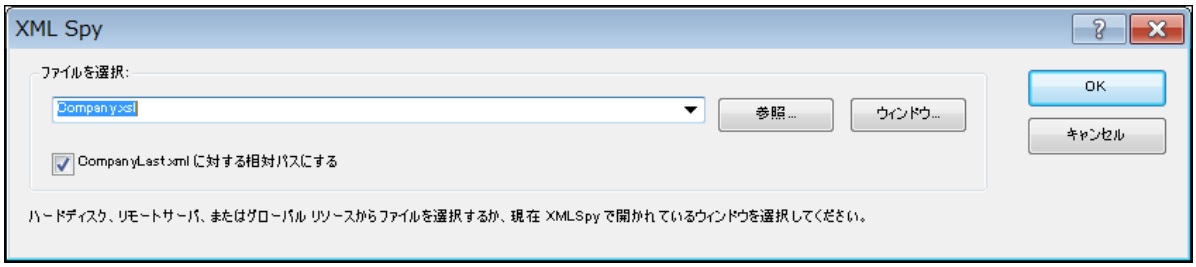
注意: XMLSpy は 1.0、2.0、および 3.0 XSLT エンジンを搭載しています。正しいエンジンは `xsl:stylesheet` か `xsl:transform` 要素のバージョン情報をもとに、XMLSpy により自動的に選択されます。このチュートリアルの変換では、XSLT 3.0 スタイルシートが使用されます。XSL 変換 コマンドが起動されたときには、変換のための XSLT 3.0 エンジンが自動的に選択されます。

2.6.1 XSLT ファイルの割り当て

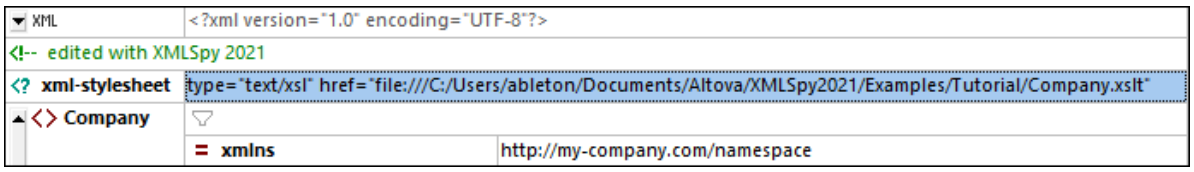
`CompanyLast.xml` ファイルに XSLT ファイルを割り当てるには:

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

(C) 2015-2021 Altova GmbH



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



2.6.2　XMLファイルの変換

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

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

Your Company

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

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

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

2.6.3 XSLファイルの修正

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

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

(C) 2015-2021 Altova GmbH

```
6   <xsl:template match="/">
7      <html>
8         <head><title>Your company</title></head>
9         <body>
10           <h1><center>Your Company</center></h1>
11           <xsl:apply-templates select="//my:Address"/>
12              <table border="1" bgcolor="#ccccff">
13                 <thead align="center">
14                    <td><strong>First</strong></td>
15                    <td><strong>Last</strong></td>
16                    <td><strong>Ext.</strong></td>
17                    <td><strong>E-Mail</strong></td>
18                    <td><strong>Manager</strong></td>
19                    <td><strong>Degree</strong></td>
20                    <td><strong>Programmer</strong></td>
21                 </thead>
22                 <xsl:apply-templates select="//my:Person"/>
23              </table>
24        </body>
25     </html>
26  </xsl:template>
27
```

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

Your Company

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

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

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

2.7 プロジェクト管理

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

2.7.1 プロジェクトの利点

XML ファイルをプロジェクトにより管理すると以下のような利点があります。

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

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

更に、次の高度なプロジェクト機能を使用することができます:

- XML ファイルを[**プロジェクト | ソース管理 | ソース管理に追加**]を使用してソースコントロール下で管理することができます。詳細に関してはソース管理セクション785 を参照してください。
- ネットワーク上の外部フォルダー800 でもweb フォルダー803をプロジェクトに追加することができます。これにより検証と変換などプロジェクトフォルダーのすべての機能がネットワークまたはインターネット上のフォルダーに適用されることが許可されます。

2.7.2 プロジェクトの作成

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

新規のプロジェクトの作成

新規のプロジェクトを以下のように作成します:

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

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

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

プロジェクトとフォルダープロパティ

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

以下の点に注意してください:

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

プロジェクトにファイルを追加する

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

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

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

その他の役に立つコマンド

その他にも良く使われる以下のプロジェクトコマンドがあります:

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

(C) 2015-2021 Altova GmbH

2.8 おしまい

ここまでくれば、おめでとう。そしてありがとう！

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

3 インターフェイスと環境

このセクションでは以下の説明を行ないます:

- アプリケーション GUI 103
- アプリケーション環境 114

GUI のセクション 103 では、最初にGUI の概要を説明した後、各 GUI ウィンドウの詳細について解説していきます。ウィンドウならびに GUI のサイズ変更、移動、そして操作方法などについても説明します。

アプリケーション環境のセクション 114 では、ファイルの管理を行うために必要な、様々な設定の表示ならびに変更方法について説明します。アプリケーションのカスタマイズを行う場所と、その方法についても記述されます。このセクションでは、あなたのコンピューター内にインストールされている、重要なサンプルファイルやチュートリアルファイルの場所についても言及されています。セクションの最後にはAltova W eb サイトへのリンクが掲載されており、アプリケーションが持つ機能の一覧や、様々な形式のユーザーマニュアル、利用可能なサポートオプション、そして Altova が提供するその他の製品に関する情報を参照することができます。

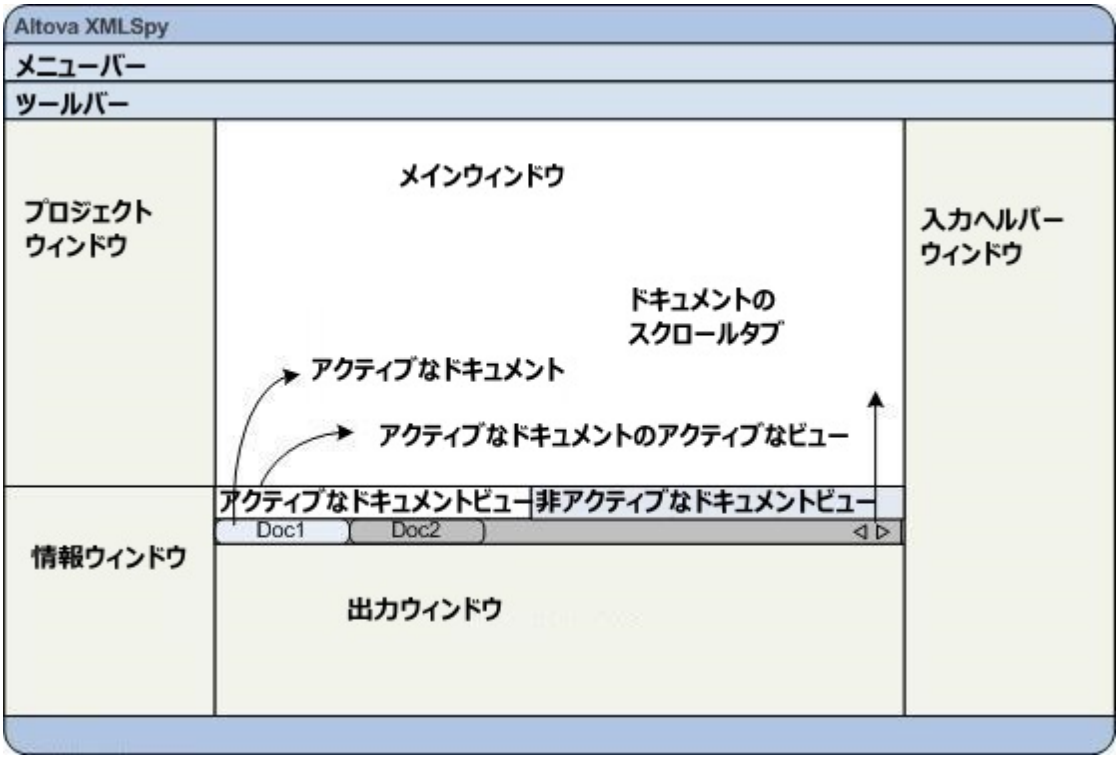
(C) 2015-2021 Altova GmbH

3.1 グラフィカルユーザインターフェース (GUI)

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

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

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



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

サイドバー表示の切り替え

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

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

サイドバーのフロートとドッキング

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

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

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

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

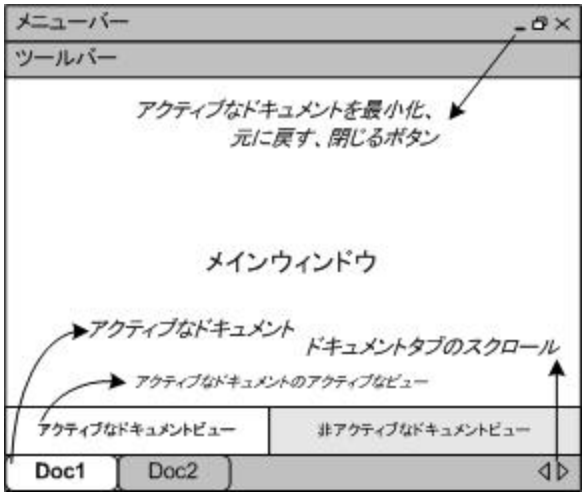
サイドバーを自動的に隠す

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

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

3.1.1 メイン ウィンドウ

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



メインウィンドウのファイル

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

- 複数のファイルが開かれている場合、ドキュメントタブバーに収まりきらず、一部のドキュメントタブが表示されないことがあります。目的のドキュメントタブは、(i) ドキュメントタブバーの右にあるスクロールボタンを使うことで、または(ii) ウィンドウ[1044] メニューから目的のドキュメントを選択することでアクティブにすることができます。
- アクティブなドキュメントが最大化されると、「**最小化**」「**元に戻す**」、「**閉じる**」ボタンがメニューバーの右側に配置されます。ドキュメントが重ねて表示、または最小化されると、「**最大化**」、「**元に戻す**」、「**閉じる**」ボタンがドキュメントウィンドウのタイトルバーに配置されます。
- 1つのファイルを最大化すると、開かれている他のファイルも全て最大化されます
- ウィンドウ[1044] メニューのコマンドを使用することで、開かれているドキュメントを重ねて、または上下(左右)に並べて表示することができます。
- 開かれているファイルは、「**Ctrl+Tab**」または「**Ctrl+F6**」により、開かれた順序でアクティブに切り替えることができます。
- ドキュメントタブを右クリックすることで表示されるコンテキストメニューから、「**印刷**」や「**閉じる**」といったコマンドを実行することができます。

メインウィンドウ内のビュー

アクティブなドキュメントを表示、編集する方法は何通りかあります。使用することのできるビューは、(上の図にあるように)ドキュメントタブ上のバーに表示されており、アクティブなビューがハイライトされます。目的のビューボタンをクリック、または表示[1044] メニューのコマンドを使用することでビューの切り替えを行うことができます。

利用可能なビューは編集またはブラウザビューです:

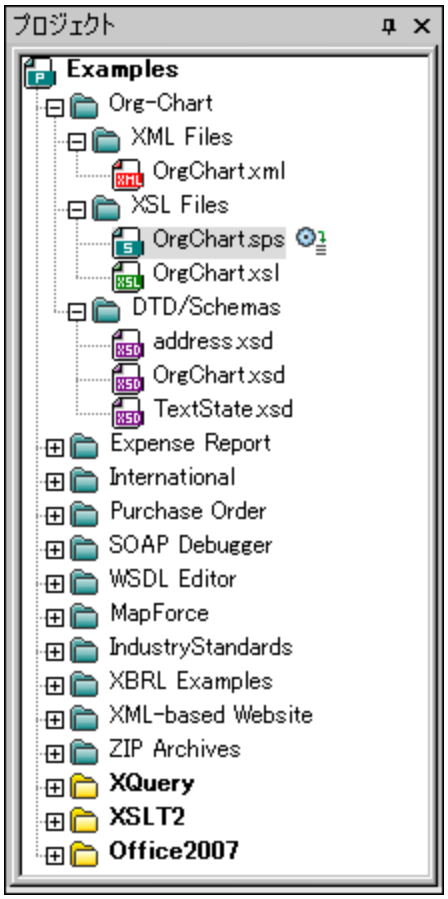
- テキストビュー[122]: 構文による色分けなどの機能を備えたソースコードレベルの編集を行うビュー
- グリッドビュー[141]: 構造化された編集を行うビュー。グラフィカルな編集を行うことのできる構造化されたグリッドの形式で、ドキュメントは表示されます。このビューにはテーブルビュー[143] が内蔵されており、繰り返し出現する要素をテーブル形式で表示することができます。
- スキーマビュー[118]: XML スキーマの表示や編集を行ないます。
- Authentic ビュー[415]: StyleVision Power Stylesheet をベースにした XML ドキュメントの編集をグラフィカルなインターフェイスで行ないます。
- ブラウザビュー[224]: CSS ならびに XSL スタイルシートをサポートする統合されたブラウザビューです。

メモ 各ファイル拡張子のデフォルトビューは、「ツール | オプション[1014]」ダイアログの、ファイルタブにあるデフォルトビューペインにてカスタマイズすることができます。

3.1.2 プロジェクト ウィンドウ

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

メモ サンプルプロジェクトはデフォルトでプロジェクトウィンドウ内で開かれます。開く必要がある場合、アプリケーションのドキュメントフォルダー[27] 内の**`Examples`** フォルダーに移動してファイル**`Examples.spp`** をダブルクリックします。



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

プロジェクトの操作

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

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

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

更に詳細なプロジェクトの使用方法に関しては、プロジェクト 676 のセクションを参照ください。

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

3.1.3 情報ウィンドウ

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

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

以下の点に注意してください:

- XSLT ファイルがアクティブな場合、情報ウィンドウの XSLT タブには XSLT 固有の情報やコマンドが表示されます(以下のスクリーンショットの XSLT タブを参照)。XSLT タブに表示される情報やコマンドの読み方については、XSLT と XQuery | XSLT | XSL アウトライン 329 のセクションを参照ください。
- JSON ドキュメント (インスタンスまたはスキーマ) がアクティブな場合、情報ウィンドウの JSON タブ内で返照とインテリジェントな編集のためのオプションを使用することができます。詳細に関しては JSON ドキュメントの検証 537 を参照してください。
- XULE ドキュメントがアクティブなドキュメントの場合、情報ウィンドウの XULE タブが XBRL インスタンスの構造に関する情報のために使用する XBRL タクソノミを選択するオプションを提供します。タクソノミ情報は XULE エディターの自動補完機能のために使用されます。

3.1.4 入力ヘルパー

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

利用可能な入力ヘルパーは以下の条件により変化します:

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

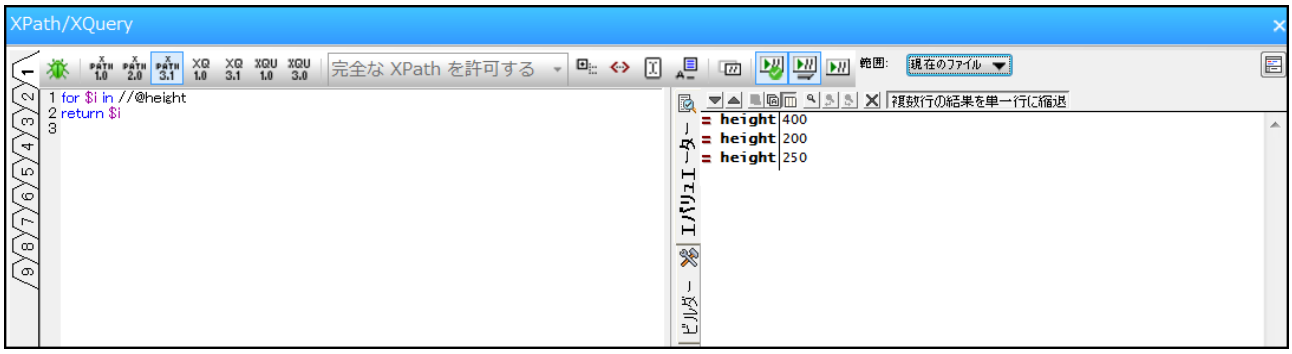
各ビューにおける入力ヘルパーの詳細については編集ビュー 118のセクションを参照ください。異なるドキュメントの種類と入力ヘルパーの関係については、(例えば、XML 入力ヘルパー 270 やXQuery 入力ヘルパー 335といった) 各ドキュメントの記述を参照ください。

以下の点に留意してください:

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

3.1.5 出力ウィンドウ: XPath/XQuery

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



このセクションは XPath/XQuery ウィンドウの主要な機能の概要を説明しています。XPath/XQuery ウィンドウとの作業方法に関する詳細は、XPath/XQuery 式 381セクションを参照してください。

主要な機能

XPath/XQuery ウィンドウには以下の主要機能が搭載されています:

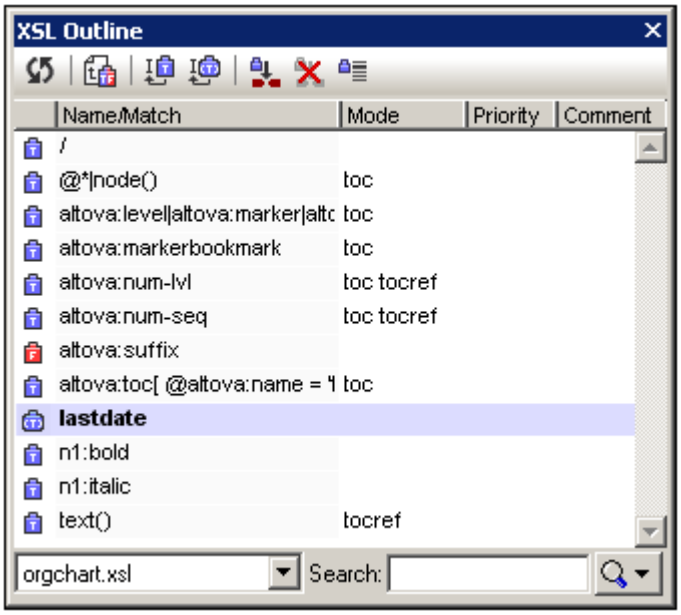
- *評価モード*: 評価モード内では、XPath または XQuery 式は 1 つまたは複数の XML/JSON ドキュメントに対応して評価されます。式は式ペインに入力され、横の結果ペイン内で結果が表示されます。結果内のノードをクリックして、XML または JSON ドキュメント内でそのノードに移動することができます。

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

XPath/XQuery ウィンドウ機能の詳細に関しては、XPath/XQuery 式 381 を参照してください。

3.1.6 出力ウィンドウ : XSL アウトライン

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

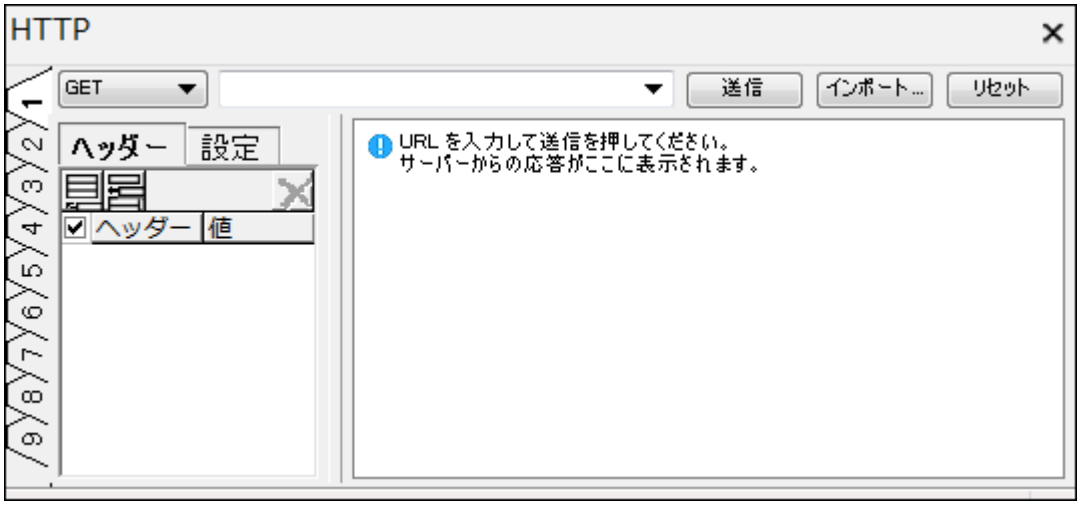


XSL アウトラインウィンドウでは、スタイルシート内のテンプレートや関数に関する情報が表示されます。ウィンドウでは、これら情報の並べ替えや検索を行い、ツールバーにあるコマンドを使用することで名前付きテンプレートに呼び出しを挿入し、名前付きテンプレートを変換の開始点にセットすることができます。XSL アウトラインウィンドウの使用方法に関しては、XSLT とXQuery | XSLT | XSL アウトライン | XSL アウトラインウィンドウ 326 のセクションを参照ください。

メモ スタイルシートのファイルに関連した情報やコマンドは、Info ウィンドウの XSLT タブから使用することができます。これらコマンドの使用方法については、XSLT とXQuery | XSLT | XSL アウトライン | Info ウィンドウ[329] のセクションを参照ください。

3.1.7 出力ウィンドウ: HTTP

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



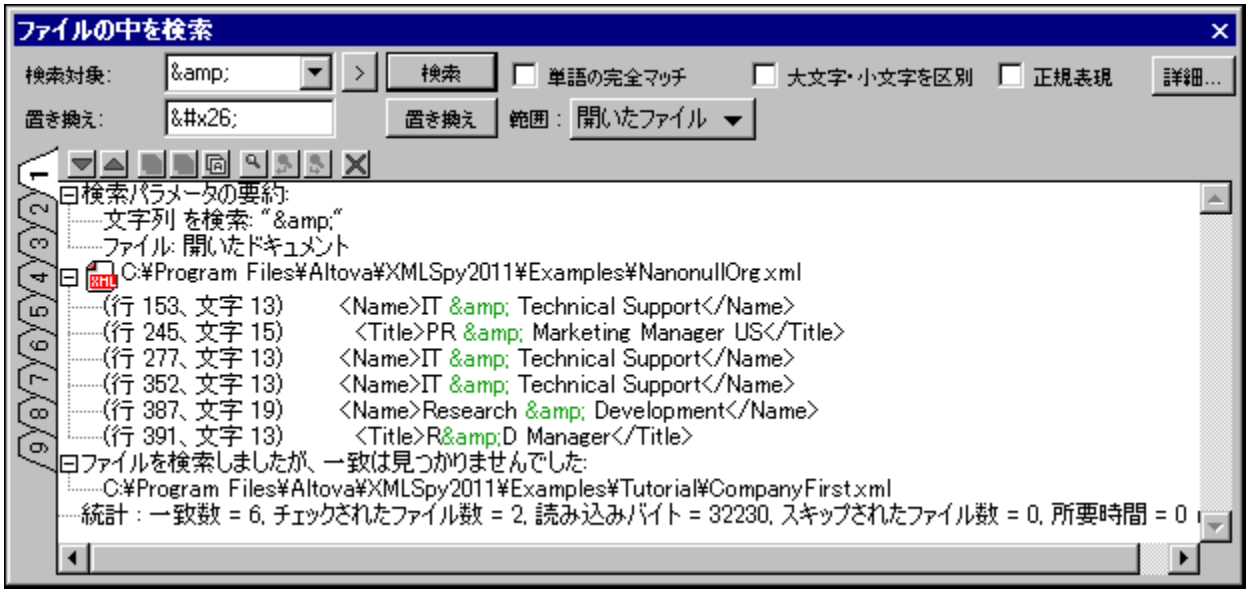
ウィンドウは以下のパートにより構成されています:

- 上の部分: (i) 使用するHTTP メソッドを選択するコンボボックス。(ii) W eb サーバーのURL のための入力フィールド。(iii) HTTP リクエストの実行に関連したボタン (「**送信**」、「**インポート**」、および「**リセット**」)。
- リクエストの作成[551] のための左側のペイン
- レスポンスの表示[557] のための右側のペイン

HTTP 出力ウィンドウの使用方法に関しては、HTTP[550] のセクションを参照してください。

3.1.8 出力ウィンドウ : ファイルの中を検索

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



検索条件

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

検索する対象 : 検索を行う文字列は、検索対象テキストボックスに入力します。検索の対象が、単語として完全に一致する必要がある場合、単語の完全マッチチェックボックスにチェックを入れます。例えば、fit という検索文字列に対して単語の完全一致オプションを有効にすると、fit という単語に対してだけ検索が行われ、例えばfitness という単語は検索結果に含まれません。検索において大文字・小文字の区別をしたい場合は、大文字・小文字を区別チェックボックスにチェックを入れます。検索テキストボックスに入力された検索文字列が正規表現となる場合、正規表現チェックボックスにチェックを入れる必要があります。 > ボタンをクリックすることで、正規表現の入力を簡単に行うことができるようになります。検索における正規表現の使用方法に関しては、検索[773]のセクションで説明されます。詳細ボタンをクリックすることで、ファイルの中を検索[778]が表示され、検索における高度な条件やアクションをセットすることができます。詳細については、編集 | ファイルの中を検索[778] を参照ください。

検索範囲 : 検索は、(i) GUI 上に開かれている全てのファイル、(ii) 現在のプロジェクトにあるファイル、または(iii) 選択されたフォルダーのファイルに対して行うことができます。詳細ボタンをクリックすることで表示されるファイルの中を検索ダイアログ[778]で、更に詳細な条件を追加することができます。

置き換え

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

結果

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

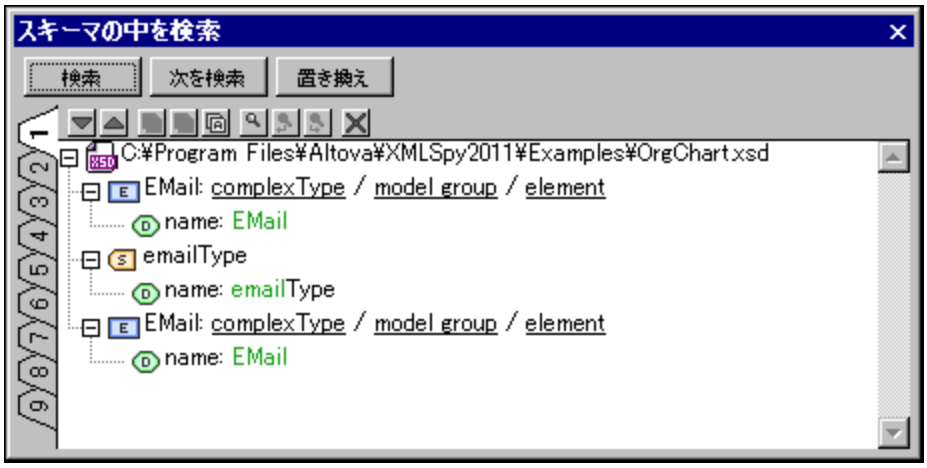
- 検索文字列や検索対象となった範囲を表示する、検索パラメータの要約。
- 検索または置き換えられた文字列のリスト(検索、または「置き換え」ボタンが押下により変化)。このリストに表示されるアイテムは、検索または置き換えがなされた箇所へのリンクとなり、クリックすることでドキュメントの該当箇所がテキストビューで表示されます。ドキュメントが開かれていない場合、そのドキュメントがテキストビューで開かれ、該当箇所がハイライトされます。

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

メモ ファイルの中を検索機能では、複数のファイルに対して、一度に**検索**や**置き換え**コマンドを実行し、その結果を出力ウィンドウに表示します。結果を逐次確認しながら進んでいく検索を行うには、検索[773]コマンドを使用してください。

3.1.9 出力ウィンドウ：スキーマの中を検索

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



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

結果は (1から9と番号の振られた) 9つあるタブに表示されます。行われた検索の結果を1つのタブに保持しておき、新たな検索の結果を別のタブにて行うことで、検索結果を比較することができます。検索を行う前に新しいタブを選択することで、新たなタブにて検索が行われます。スキーマの中を検索ウィンドウ内に表示される結果をクリックすることで、メインウィンドウのスキーマビューにて、該当する箇所が表示、ハイライトされます。出力タブに表示されるメッセージはクリップボードにコピーすることもできます。詳細については、結果と情報[317] 機能の、スキーマ内の検索[307] のセクションを参照ください。

3.1.10 メニューバー、ツールバー、ステータスバー

メニューバー

メニューバー (*図を参照*[103]) には、様々なアプリケーションメニューが収められています。以下の慣習が適用されます:

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

- いくつかのメニューコマンドはサブメニューを持っており、更に別のオプションを選択することができます。サブメニューがあるメニューコマンドは、コマンド名の右側に表示される右方向矢印により示されます。
- メニューコマンドにアクセスするには、メニュー名をクリックした後に、コマンドを選択します。メニューアイテムにサブメニューがある場合、そのメニューアイテムにマウスを移動することで、サブメニューが開かれます。その後、目的のサブメニューアイテムをクリックします。
- 適切なキーのコンビネーションにより、キーボードからメニューを開くこともできます。各メニューへのキーコンビネーションは **Alt+*KEY*** で、***KEY*** がメニュー名のキーになります。例えば、「**ファイル(F)**」メニューへのアクセスを行うには、**Alt+F** を押下します。
- (i) メニューに対するキーの組み合わせ(前の項目を参照)の後に、(ii) 特定のコマンドに対するキーの組み合わせを入力することで、(メニュー内に収められているコマンドの) メニューコマンドへアクセスすることができます。例えば、新規ファイルを作成(「**ファイル(F) | 新規作成 (N)**」)するには、**Alt+F** を押下した後、**Alt+N** を押下します。
- 特定のメニューコマンドは、**ショートカットキー**または、キーのコンビネーション(**Ctrl+*Key***)により直接選択することができます。ショートカットが割り当てられたコマンドは、コマンド名の右側にショートカットキーや、キーのコンビネーションが示されます。例えば、ショートカットキーのコンビネーションである **Ctrl+N** を使用することで、ファイルの新規作成を行うことができ、**F8** によりXML ファイルの検証を行うことができます。カスタマイズダイアログ(「**ツール | カスタマイズ**」)にあるキーボードタブにて、独自のショートカットを作成[1003] することもできます

ツールバー

メニューバー(*図を参照*[103]) には、メニューコマンドを選択するためのショートカットとなるアイコンが収められています。マウスポインターをアイコン上に移動させると、コマンドの名前が表示されます。アイコンをクリックすると、コマンドが実行されます。

ツールバーボタンはグループでまとめられています。「ツール | カスタマイズ | ツールバー[1000]」ダイアログでは、表示するツールバーのグループを選択することができます。これらの設定は現在のビューに対して適用されます。他のビューの設定を変更するには、そのビューに切り替えた後、「ツール | カスタマイズ | ツールバー[1000]」にて設定を変更します。GUI では、ツールバーグループのハンドル(またはタイトルバー)を別の位置にドラッグすることで、ツールバーグループの位置を変更することができます。ハンドルをダブルクリックすると、ツールバーがドッキングからフロート状態になり、タイトルバーをダブルクリックすることで、元の場所にドッキングされます。

ステータスバー

ステータスバーはアプリケーションウィンドウの下部に位置(*図を参照*[103])しており、(i) ファイルのロードに関する情報や、(ii) マウスポインターがメニューコマンドや、コマンドショートカットにある時に、それらコマンドに関する情報を表示します。64ビット版のXMLSpyを使用している場合、XMLSpy の後に (x64) という表記が加わります。この表記は32ビットバージョンにおいてはなされません。

3.2 アプリケーション環境

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

このセクションは以下のように構成されます:

- *設定、カスタマイズ*[114]: 重要な設定やカスタマイズを行う場所と、その変更方法について説明します。
- *チュートリアル、プロジェクト、サンプル*[116] アプリケーションパッケージに収められている、様々な非プログラムファイルの場所と、その注釈について記述します。
- *製品機能とドキュメンテーション、Altova 製品*[117]: : 製品の機能や、その他の形式で記述されたヘルプ、その他の Altova 製品に関する情報が記述されたAltova W eb サイトへのリンクが掲載されます。

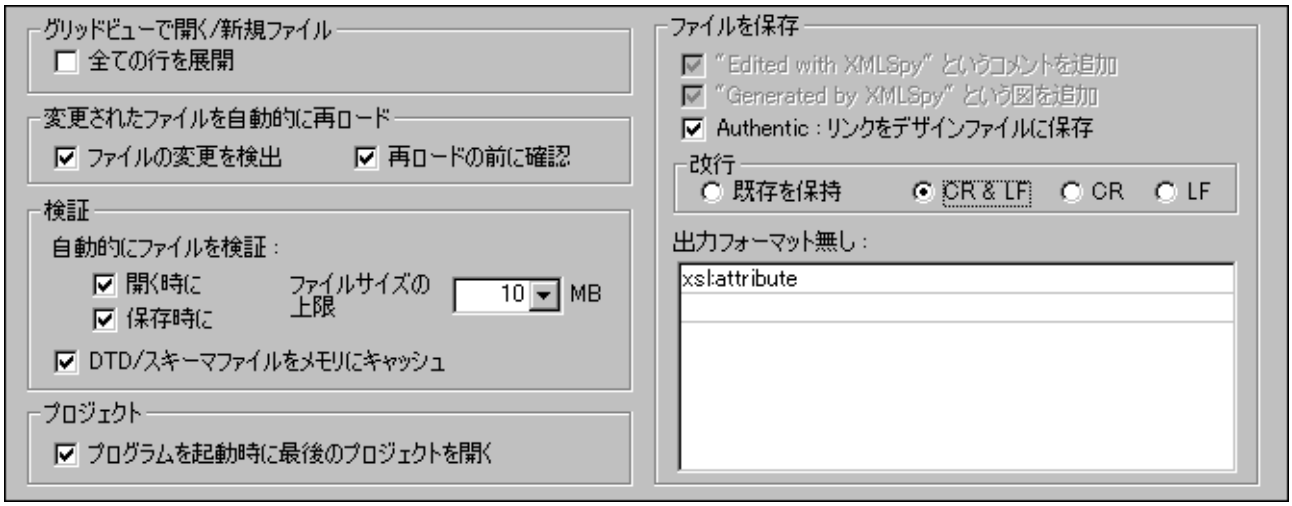
3.2.1 設定、カスタマイズ

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

- 設定[114]
- カスタマイズ[115]
- メニュー[115]

設定

オプションダイアログのタブには、XMLSpy が持つ重要な設定が定義されています（以下のスクリーンショットはメニューコマンド「ツール | オプション[1014]」によりアクセスすることができます）利用可能なオプションがどのようなものか確認してください。

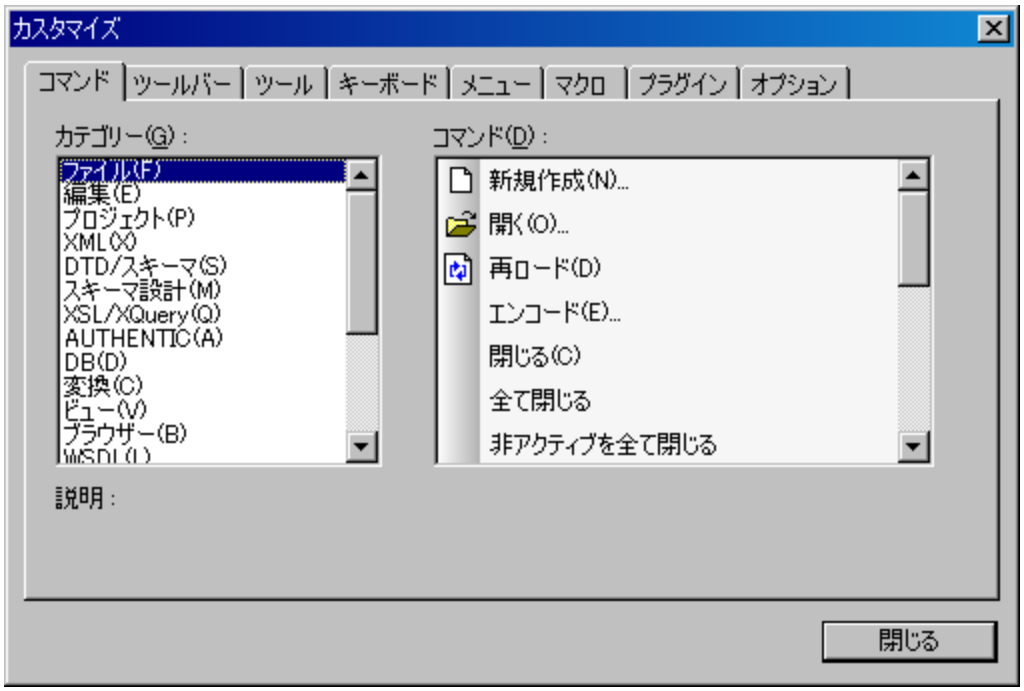


以下は最も重要な設定の要約です。詳細については、ユーザリファレンスにある オプションダイアログ[1014] の記述を参照ください。

- *ファイルタイプとデフォルトビュー* : ファイルタイプタブでは、XMLSpy が認識するファイルの種類を追加することができます。新たなファイル拡張子をリストの中に追加してください。各ファイルタイプに対して、そのファイルが準拠する仕様や、その種類（拡張子）のファイルを開く際に使用される XMLSpy のビュー（デフォルトビュー）を指定します。
- *ファイルの検証* : ファイルタブでは、ファイルを開く、保存する際に、ファイルの検証を自動的に行うか設定することができます。ファイルタイプタブでは、指定したファイルの種類の検証を無効にすることができます。
- *編集機能* : 編集タブでは、入力ヘルパーの管理方法や、新規要素の生成、更に自動補完を有効にするなどの設定を行うことができます。ビュータブにおいても、その他の設定を個別に行うことができます。各ビューに対するフォントタブでは、これらビュー内部におけるノードの各種類のフォント設定を行うことができます。
- *XSLT と FO エンジン* : XSL タブでは、GUI から行われた変換で使われる外部 XSL エンジンを指定することができます。XMLSpy. 内部から FO 処理を行う際に使用される FO プロセッサーの場所も指定する必要があります。詳細については、XSLT 処理 323 のセクションを参照ください。
- *エンコーディング* : XML ならびに非 XML ファイルのエンコーディングはエンコードタブにより指定されます。

カスタマイズ

ユーザーは GUI の外観など、XMLSpy の様々な側面をカスタマイズすることができます。これらのカスタマイズオプションは、カスタマイズダイアログからアクセスすることができます（以下のスクリーンショットを参照）メニューコマンド「ツール | カスタマイズ 998 」からアクセスすることができます）。



カスタマイズコマンドに関する詳細はユーザーレファレンス 998 のセクションを参照ください。

メニュー

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

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

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

メモ: メニューのカスタマイズ（「ツール | カスタマイズ 998」）では、アイテムを GUI にドラッグすることで、コマンドの再構成を行うことができます（ メニューバー、ツールバーステータスバー 112 を参照）。

3.2.2 チュートリアル、プロジェクト、サンプル

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

チュートリアル、プロジェクトサンプルファイルの場所

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

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

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

マスター XMLSpy フォルダーに関する注意

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

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

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

チュートリアル、プロジェクト、サンプルファイルの場所

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

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

3.2.3 XMLSpy 製品機能と Altova 製品

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

XMLSpy 機能の一覧

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

XMLSpy ヘルプ

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

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

サポートオプション

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

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

Altova 製品

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

4 編集ビュー

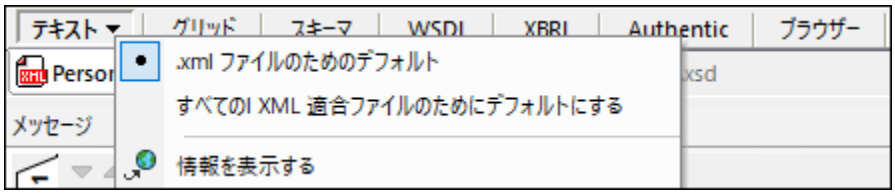
XMLSpy にはパワフルな編集ビューがあります。インテリジェント編集機能を備えたテキストビューに加えて、ドキュメントの編集を飛躍的に簡易化して表示するグラフィカルなビューも備わっています。XMLSpy にてアクティブになっているドキュメントの種類により、メインウィンドウには1つまたは複数の編集ビューが表示されます。例えば、HTML ドキュメントがアクティブの場合、メインウィンドウにはテキストビューとブラウザービューの2つの編集ビューが表示されます。XMLドキュメントがアクティブな場合、5つの編集ビュー(テキストビュー、グリッドビュー、スキーマビュー、Authentic ビュー、そしてブラウザービュー)が表示されます。これらのビューの中でスキーマビューはXML ドキュメントに対してのみ有効になります。

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

- テキストビュー 122
- グリッドビュー 141
- スキーマビュー 151
- Authentic ビュー 223
- ブラウザービュー 224

デフォルトビューの選択

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



メニューオプションは以下の通りです:

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

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

以下の場合、デフォルト編集ビューの変更は不可能です。

- *JSON スキーマファイル:* JSON ファイルのようなこれらのファイルは、`.json` 拡張子を持ちます。しかしながら、自動的にJSON スキーマのコンテンツから検知され、常にJSON Schema View 505 で開かれます。ます。
- *Authentic-が有効化されたXML ファイル:* これらはXML ドキュメントをAuthentic XML ドキュメントとしてビューすることを可能にするAltova StyleVision スタイルシートを割り当てられた(`.xml` ファイル拡張子を持つ) XML ファイルです。 これらのファイルはAuthentic View 223 で開かれます。

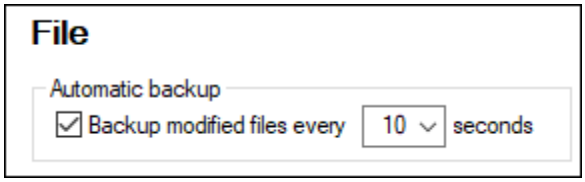
(C) 2015-2021 Altova GmbH

メモ ファイル拡張子のファイルコンポーネント、および各ファイル型に対応するデフォルトの編集ビューのファイルの適合性は、オプションダイアログのファイル型セクション[1018]内で設定することができます。しかしながら、（ここで説明されているとおり）編集ビュー自身内でのデフォルトビューの選択により他の方法よりも早い方法でデフォルトビューを設定することができます。

4.1 ファイルの自動的なバックアップ

XMLSpy 内で変更されたファイルは定期的な頻度で自動的にバックアップされます。(「ツール | オプション | ファイル」[1015])オプションダイアログの *ファイル* タブで以下のスクリーンショットで示されている通り以下を行うことができます:

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



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

インジケーター

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

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

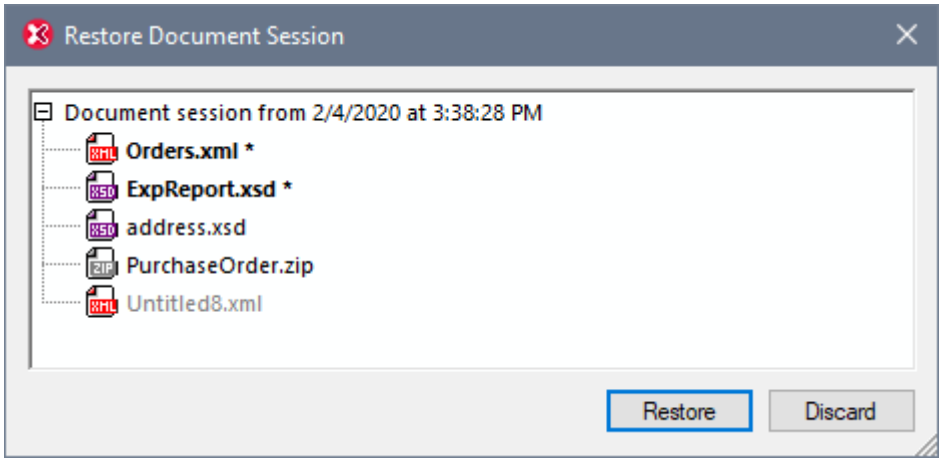
バックアップの状態
サークルのシンボルはファイルのバックアップの状態を示しています。

- *黄色:* ファイルが変更済みで、最後の変更時からバックアップ(または未保存)の場合。
- *緑:* ファイルがバックアップ済みで、最後のバックアップ時から変更されていない場合。しかしながら、ファイルが保存されていない場合。(保存されている場合、赤いサークルのシンボルは表示されません。)
- *赤:* このファイルのためのバックアップはサポートされない場合(例えば、ZIP ファイルの アーカイブビュー[225] など) 。またはバックアップに失敗した場合。
- *グレー:* 自動的なバックアップ機能が無効化されている場合(via the オプションダイアログ[1015]の使用。*上記を参照してください*)。このシンボルの存在はしかしながらファイルが最後の変更時から保存されていないことを意味します。(保存されている場合、赤いサークルのシンボルは表示されません。)

バックアップの復元

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

(C) 2015-2021 Altova GmbH



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

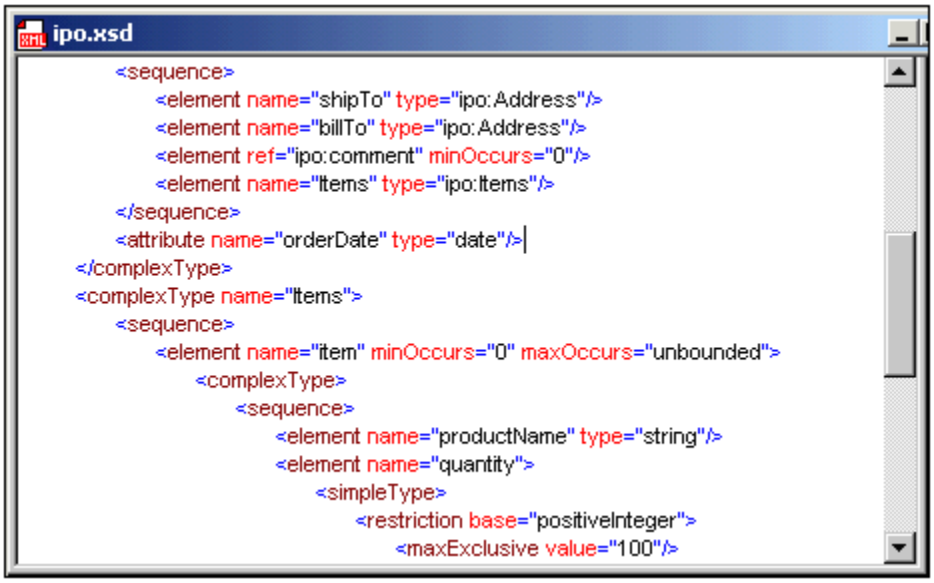
- 太字とアスタリスクはファイルに保存されていない変更が含まれていることを示しています。このようなファイルは最後にバックアップされた状態で復元されます。
- 標準のスタイルはファイルが保存済みで未保存のファイルが存在しないことを示しています。このようなファイルは保存された状態で復元されます。
- 灰色表示されているスタイルはファイルが保存されておらず、またバックアップされていないことを表示しています（例えば、編集されていない新規のファイルなどが挙げられます）。このようなファイルは復元されません。

以下の一つを行うことができます:

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

4.2 テキストビュー

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



このセクションでは、全ての種類のドキュメントで使用することのできる一般的なテキストビューの機能を紹介します。XMLやXQuery、そしてCSS といった特定の種類のドキュメントに対して割り当てられている機能については、それらのドキュメントタイプに関するセクションにて記述されます。例えば、テキストビューにおけるXML 独自の機能についてはXML | テキストビューでXML を編集する 234 セクションで記述されます。

一般的なテキストビューの機能は以下のように分類されています:

- テキストビューでのフォーマット 123 ドキュメントのフォントプロパティ、インデンテーション、そしてワードラップの設定方法を説明します。
- ドキュメント表示 124 行番号、ブックマーク、ノードの展開/折りたたみ、そしてその他の表示に関わる機能についての情報を提供します。
- テキストビューでの編集 127 編集中に使用できる機能、特にインテリジェント編集機能について説明します。
- ドキュメント内の移動 129 テキストビュー内でドキュメントをナビゲートする様々な方法について説明しています。
- 検索および置換 132 テキストビューの検索および置換機能について説明しています。
- 入力ヘルパー 137 コンテキストにより変化するデータ入力オプションを提供するウィンドウです。例えば、指定された位置で入力することのできる要素または属性が入力ヘルパーに表示され、ダブルクリックすることによりオプションを挿入することができます。
- 分割ビュー 138 はテキストビューのメインウィンドウを2つに分割し、あくまでもドキュメントを両方のビューで表示します。これにより、長いドキュメントのところなるパートを横に並べて確認することができます。
- テキストビューショートカット 140 はテキストビューコマンドにより一般的に使用されるデフォルトのショートカットをリストしています。

テキストビューに切り替える

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

4.2.1 テキストビューでのフォーマット

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

フォント

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

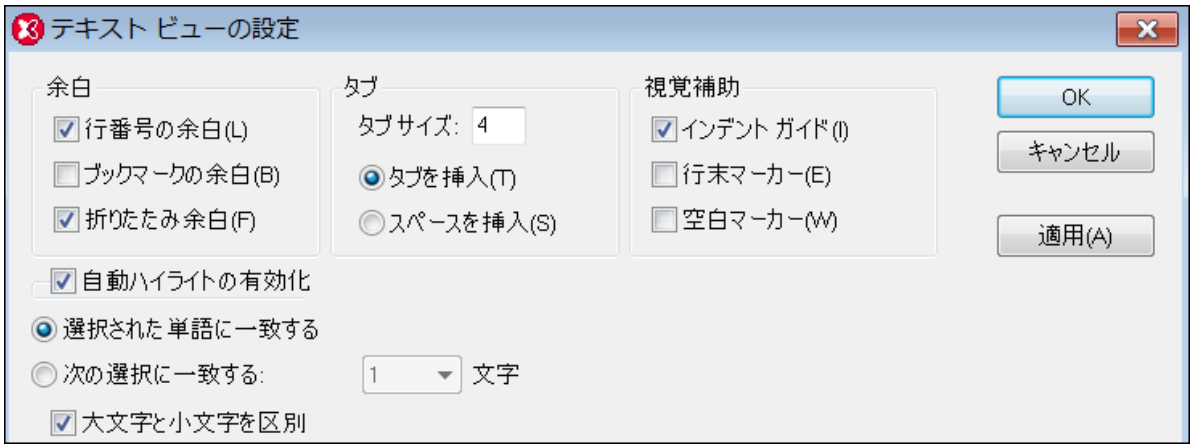
異なる意味を持つドキュメント内のテキストアイテムには別々の色を与えることができます。例えば、要素の名前、属性の名前、そして要素の内容を別々に色付けすることができます。テキストアイテムに対して異なる色付けをした場合、構文のカラー強調表示機能が有効になります。オプションダイアログのフォントと色のセクション 1029 内でテキストフォントをカスタム化することができます。

インデント

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

ドキュメントをこの方法で表示するには、以下を行う必要があります:

1. オプションダイアログのビューセクション内で、整形出力のためのインデントの使用をチェックします。これにより階層構造を示すためにドキュメントがインデント共に整形出力されます。更に深いレベルは親要素より先深いインデントにより表示されます。インデントの使用オプションがチェックされていない場合、ドキュメント内のすべてのラインはインデントがゼロの箇所から開始します。
2. テキストビューの設定ダイアログ(「ビュー | テキストビュー設定 969」*下のスクリーンショットを参照*)にて、「タブを挿入」か「スペースを挿入」を選択します。これによりドキュメントが整形出力される場合インデントにタブまたはスペースが使われるのかが決定されます。スペースが指定されている場合、階層のより深い箇所に、テキストビュー設定のタブサイズ内で指定されている数だけのスペースが挿入されインデントを形成します。



3. 「編集 | 整形出力」[771] コマンドを選択するか、ツールバーの整形出力アイコンをクリックします。これによりドキュメントテキストがオプションダイアログの整形出力[1022] セクション内で指定されているフォーマットに従い表示されます。整形出力コマンドは不必要な先頭または末尾の空白を削除することに注意してください。

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

フォーマットにタブとスペースを使用する

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

ワードラップ

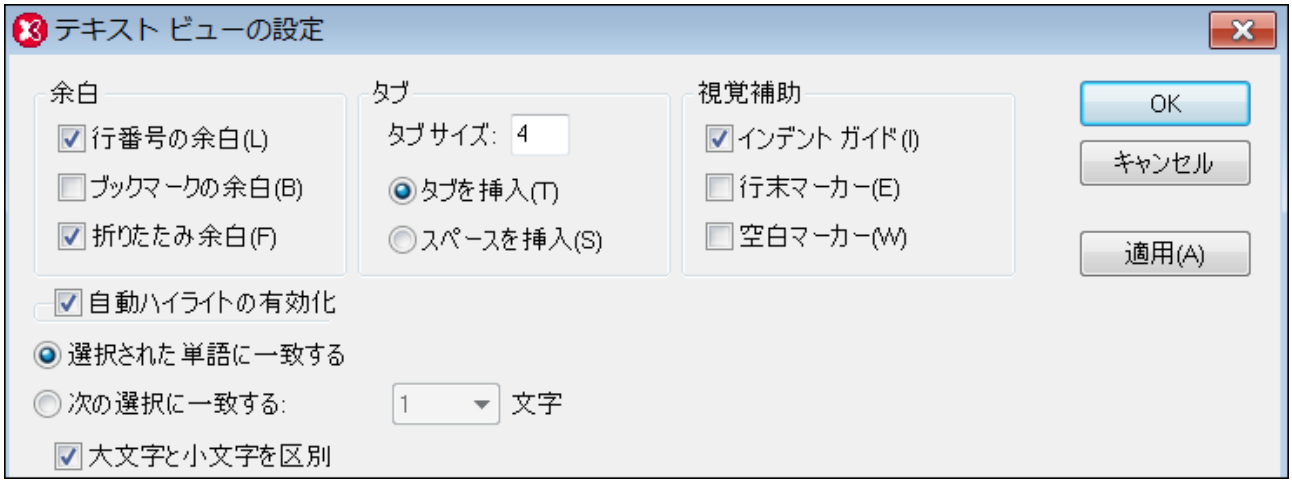
「表示 | ワードラップ」[968] コマンドを有効にすることで、メインウィンドウの幅より長い行は折り返して表示されます。対応するアイコンはテキストツールバー[124] にもあります。

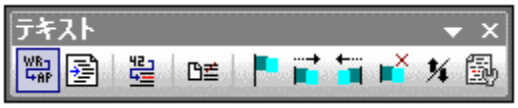
4.2.2 ドキュメントの表示

テキストビューでは大規模なテキストの閲覧と編集を簡単にする視覚機能を使用することができます。とても便利な機能として: (i) 行番号[125]、(ii) ブックマーク[125]、(iii) ソースの折りたたみ[125]（ノードごとに展開し折りたたんだりする機能）、(iv) インデントガイド[126]、そして (v) 行末マーカーと空白マーカー[126] があります。これらのコマンドはテキストビュー設定ダイアログ（1番目のスクリーンショット）とテキストツールバー（2番目のスクリーンショットを参照）にて使用することができます。

テキストビュー設定ダイアログ[969] には「**表示 | テキストビューの設定**」コマンドまたは、テキストツールバーの「**テキストビューの設定**」ボタンからアクセスすることができます。テキストビュー設定ダイアログの設定内容はアクティブなドキュメントだけでなく、アプリケーション全体に対して適用されます。

(C) 2015-2021 Altova GmbH





その他の便利な機能として「ズーム」127 と「ナビゲートと検索」129 機能があります。

行番号

行番号は行番号余白に表示され(下のスクリーンショットを参照)、テキストビュー設定ダイアログで有効/無効にすることができます(上のスクリーンショットを参照)。テキストのセクションが折りたたまれているときには、折りたたまれている行の番号も同様に隠されます。関連するコマンドに「行/文字へ移動」129 コマンドがあります。

ブックマーク

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

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

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

ソースの折りたたみ

テキストの折りたたみ(または*ソースのおりたたみ*) はXQuery 中かっこ、XQuery コメント、XML 要素、XML コメントにより有効化され、これらのノードを展開または折りたたむ機能を意味します。このようなノードは、+/- サインによりソース折りたたみ余白内で表示されます(*下のスクリーンショット参照*)。余白は*テキストビュー設定ダイアログ*969 により切り替えのオンとオフを行うことができます(*上のスクリーンショット参照*)。下のスクリーンショット内では、3つのノードが折りたたまれています: `shipTo` 要素2つの `item` 要素。ノードが折りたたまれると、視覚的に省略記号により表示されます(下のスクリーンショットでは緑の枠でかこまれています)。内容がポップアップでの表示に大きすぎる場合、ポップアップのしたの部分に省略記号で表示されます。

```
1   <?xml version="1.0"?>
2   <!-- edited with XMLSPY v2004 rel. 4 U (http://www.xmlspy.com) by Mr. Nobody
    (Altova GmbH) -->
3   <ipo:purchaseOrder xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:ipo="http://www.altova.com/IPO" orderDate="1999-12-01"
    xsi:schemaLocation="http://www.altova.com/IPO
4   .xsd">
5       <shipTo export-code="1" xsi:type="ipo:EU-Address">...</shipTo>
11      <billTo xsi:type="ipo:US-Address">
12          <ipo:name>Robert Smith</ipo:name>
13          <ipo:street>8 Oak Avenue</ipo:street>
14          <ipo:city>Old Town</ipo:city>
15          <ipo:state>AK</ipo:state>
16          <ipo:zip>95819</ipo:zip>
17      </billTo>
18      <Items>
19          <item partNum="833-AA">
20              <productName>Lapis necklace</productName>
21              <quantity>2</quantity>
22              <price>99.95</price>
23              <ipo:comment>Need this for the holidays!</ipo:comment>
24              <shipDate>1999-12-05</shipDate>
25          </item>
26          <item partNum="748-OT">...</item>
33          <item partNum="783-KL">...</item>
39      </Items>
40  </ipo:purchaseOrder>
41
```

```
<productName>Uncut diamond</productName>
<quantity>7</quantity>
<price>79.90</price>
<shipDate>2000-01-07</shipDate>
```

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

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

インデントガイド

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

行末マーカー、空白マーカー

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

```
5 ···<CompanyLogo·href="nanonull.gif"/> EOL
6 ···<Name>Organization·Chart</Name> EOL
7 ···<Office> EOL
8 ·····<Name>Nanonull,·Inc.</Name> EOL
9 ·····<Desc> EOL
```

ズームインとアウト

「Ctrl」キーを押しながらスクロール(マウスのホイール)を使用することでテキストビューの拡大と縮小を行うことができます。この機能を使うことで、テキストビューのテキストサイズを調整することができます。フォントサイズを変更したい場合は、オプションダイアログ[123]にて行うことができます。

4.2.3 テキストビューでの編集

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

- 構文の色分け[127]
- インテリジェント編集[127]
- 自動補完[128]
- 開始タグと終了タグのマッチ[128]
- 同じ階層で要素を移動させる[129]
- 要素全体を選択し親要素に移動する[129]
- ドラッグアンドドロップとコンテキストメニュー[129]
- 制限無しに元の状態に戻す[129]
- スペルチェック[129]

(XML[229] やXQuery[332] などの)ドキュメントの種類によっては、更に独自の機能が利用可能です。これらの機能については、それぞれ対応するドキュメントの種類に関するセクションで記述されています。

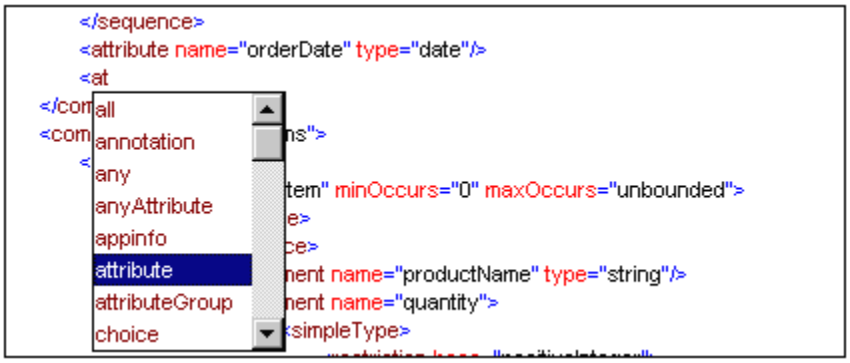
メモ　大きなファイルに関しては、速いロードと編集のため自動補完と入力ヘルパーが無効化されます。ファイルのサイズの敷居はユーザーにより指定することができます。更に詳しい情報に関しては、「オプション | 編集」[1021]を参照してください。

構文の色分け

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

インテリジェントな編集

スキーマをベースにしたXMLドキュメントを扱っている場合、XMLSpy のテキストビューで様々なインテリジェント編集機能を使用することができます。これらの機能を使用することで、現在編集しているコンテンツモデルに従った正しい要素、属性、または属性の値を素早く入力することができます。例えば、要素の開始タグの入力を開始すると、この時点でスキーマにより許可されている全ての要素の名前がポップアップで表示されます(下のスクリーンショットを参照)。要素名を選択し、「Enter」を押すことで開始タグにその要素名が入力されます。また、開始タグが作成された後、終了タグが自動的に追加されます(*下の自動補完[128] below を参照してください*)。



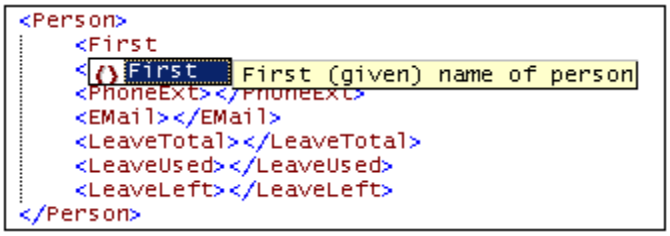
ポップアップウィンドウは以下の場合にも表示されます:

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

自動補完

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

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



自動補完はオプションダイアログの編集タブ[1021](「ツール | オプション | 編集」)にて有効/無効化することができます。

開始と終了タグの一致

カーソルがマークアップ要素の開始もしくは終了タグの中をポイントしているとき、「Ctrl」+ E を押すことで選択を他のペアのメンバーに移動することができます。「Ctrl」+E を繰り返し押すことで、開始タグと終了タグを切り替えることができます。この機能を使って、簡単にXML 要素の開始タブと終了タブを見つけることができます。これは、XML 要素のタグを開始と終了を検索するために役にたちます。更に、開始タグで、入力を始めると、開始と終了タグの名前が一致するか否かにより要素の名前は異なる色でハイライトされます[1029]。これは視覚的な編

集エイドとしての役割を果たします。色のハイライトは、オプションダイアログ[1029]で設定することができます。開始タグ内で要素の名前を編集すると、自動的に終了タグが編集されます。

互いに相対する要素を移動する

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

要素全体を選択し、親要素に移動する

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

ドラッグアンドドロップとコンテキストメニュー

ドラッグアンドドロップやテキストのブロックを他の場所に移動し、右クリックにより表示されるコンテキストメニューを使用して、(切り取り[763] やコピー[763]、ペースト[763]、削除[763]、メールで送信[757]、そして行/文字に移動[969]などの) よく使われるコマンドにアクセスすることでもテキストの移動を行うことができます。

制限無しに元の状態に戻す

XMLSpy は全ての編集オペレーションに対して制限無しに元に戻す[763] および やり直し[763] を提供します。

スペルチェック

テキストビューでは、内蔵の言語辞書を使用することでドキュメントのスペルチェックを行うことができます。ユーザー辞書を作成することで、言語処理に含まれていない単語を加えることができます。詳細については、スペリング[974] ならびにスペルチェッカーのオプション[978] コマンドを参照ください。

4.2.4　ドキュメント内の移動

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

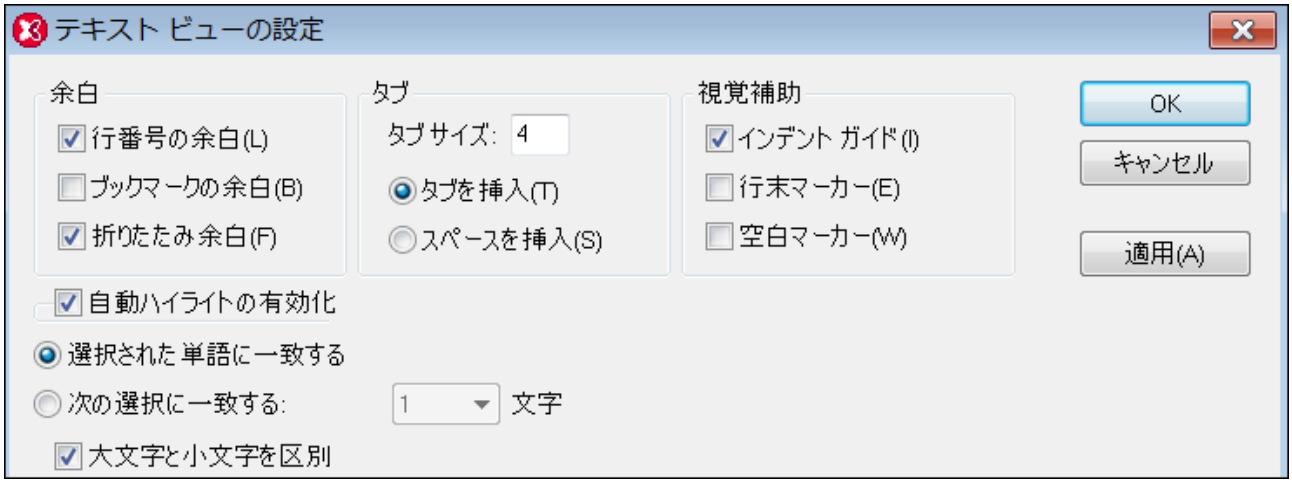
- テキストのハイライト[129] により、選択したテキスト文字列または文字に一致する全てのマッチを検索することが可能になります。それぞれの一致は、スクロールバーに示されており、全ての一致に簡単に移動することができます。
- スクロールバー内のドキュメントの概要[131] により、ドキュメント内でのカーソルとテキストの相対的な位置を表示します。
- ライン/文字に移動[131] により、指定したラインと文字に直接移動することができます。

テキストのハイライト

テキストビュー設定ダイアログ (「表示 | テキストビュー設定」[969]) 内でテキストのハイライトが有効化されていると、選択されたテキストのドキュメント内の全ての一致がハイライトされます。選択は、薄い青色でハイライトされ、一致する箇所は薄いオレンジ色でハイライトされます (*下のスクリーンショット参照*)。選択と、その一致する箇所は、スクロールバー上で四角の灰色のマーカーで表示されます。スクロールバー内で現在のカーソルの位置は、青いカーソルマーカーで示されています。

```
4
5   <expense-report xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="ExpReport.xsd" currency="USD" detailed="false"
    total-sum="556.9">
6       <Person>
7           <First>Fred</First>
8           <Last>Landis</Last>
9           <Title>Project Manager</Title>
10          <Phone>123-456-7890</Phone>
11          <Email>f.landis@nanonull.com</Email>
12      </Person>
13      <expense-item type="Lodging" expto="Sales">
14          <Date>2003-01-01</Date>
15          <expense>122.11</expense>
16      </expense-item>
17      <expense-item type="Lodging" expto="Development">
18          <Date>2003-01-02</Date>
19          <expense>122.12</expense>
20          <description>Played penny arcade</description>
21      </expense-item>
22      <expense-item type="Lodging" expto="Marketing">
23          <Date>2003-01-02</Date>
24          <expense>299.45</expense>
25          <description>Treated Clients</description>
26      </expense-item>
27      <expense-item type="Entertainment" expto="Development">
28          <Date>2003-01-02</Date>
29          <expense>13.22</expense>
30          <Misc misctype="TeamBuilding"/>
31          <description>Bought signed XMLSpy Handbook</description>
32      </expense-item>
33  </expense-report>
```

テキストのハイライトをオンに切り替えるには、テキストビュー設定ダイアログ（「表示 | テキストビューの設定」969 内の*自動ハイライトの有効化*を選択します。*下のスクリーンショット*）。選択は、文字全体、または、文字の固定された数値で定義することができます。また、キャストを考慮するかを選択することもできます。



以下の点に留意してください：

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

されます。ですから、この場合、`t` を選択すると、一致する項目は表示されません。`ty` を選択する場合、`ty` に一致するマッチが表示されます。`typ` を選択すると、`typ` に一致するすべてのマッチが表示されます。

- 文字の検索の場合、文字を区別するために次の条件が考慮されます: (角かっこのない)要素名、要素タグの角かっこ、属性名、引用符なしの属性の値。

メモ　要素開始タグと要素完了タグ[127] は、自動ハイライトの有効化に影響されない個別の機能です。

スクロールバー内のドキュメントの概要

スクロールバーには以下の機能が搭載されています:

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

以下の点に留意してください:

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

ライン/文字に移動

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

4.2.5 検索および置換

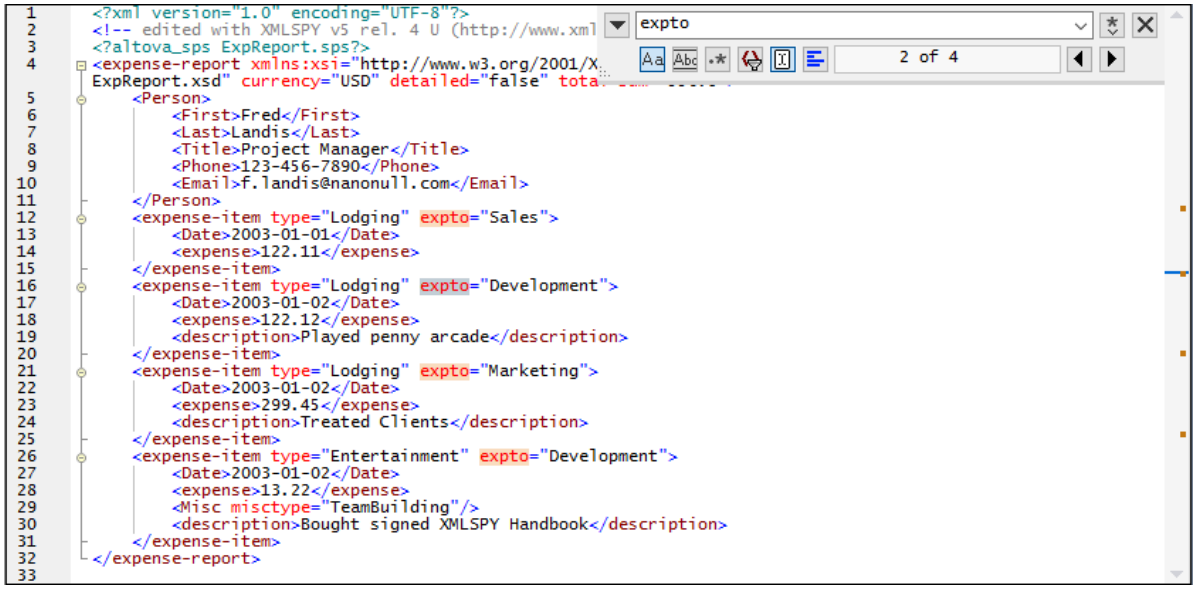
このセクション:

- ドキュメント内のテキストの検索 132
- 検索オプション 133
- テキストの置換え 133
- 正規表現の使用 134
- 正規表現のサンプル 135

ドキュメント内でのテキスト検索

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

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



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

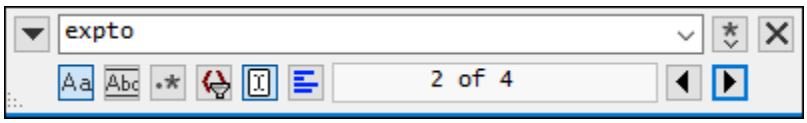
以下の点に留意してください:

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

- ダイアログボックスを開く前にテキストが選択されている場合、選択されたテキストは、自動的に検索フィールドとして既知の)検索用語フィールドに挿入されます。
- 範囲内で検索する場合、以下を行います: (i) 選択範囲をマークします。(ii) *選択範囲内での検索*[133] オプションをオンにして、選択をロックします。(iii) 検索用語を入力します。他の選択を検索するには、現在の選択を、*選択範囲内での検索*[133] オプションをオフにして、ロックを解除してください。新しい選択を決定し、*選択範囲内での検索*[133] オプションをオンにします。
- 検索ダイアログが閉じられた後、現在の検索を「 **F3** 」を押して繰り返すことができます。または、「**Shift+F3**」を押して後ろ向きに検索することができます。検索ダイアログが表示されます。

検索オプション

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



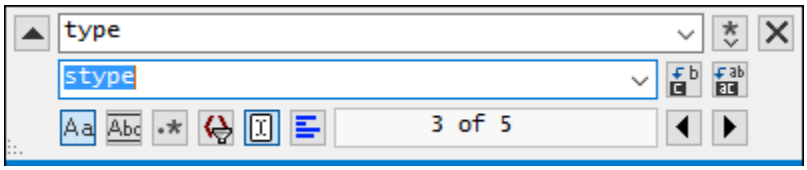
以下のオプションから選択することができます:

- *大文字と小文字を区別する*: オンに切り替える場合は大文字と小文字が区別されます (`Address` は、`address` とは異なります)。
- *単語単位*: テキスト内の単語単位が一致するものをのみを表示する。例えば、単語単位がオンに切り替えられており、入力文字列が `fit` の場合、`fit` のみが検索文字列に一致します。例えば、`fitness` 内の `fit` は、一致ではありません。
- *正規表現*: オンに切り替えられていると、検索用語は正規表現として読み込まれます。どのように正規表現が使用されるかに関する説明は、下の *正規表現*[134] を参照してください。
- *フィルターの結果*: 検索が実行される1つまたは1つ以上のドキュメントコンポーネントを選択します。
  - *アンカーの検索*: 検索用語が入力されると、ドキュメント内の一致がハイライトされ、一致の内の1つが現在の選択としてマークされます。アンカーの検索は最初の選択がカーソルに位置に相対するかを決定します。アンカーの検索がオンになっていると、最初の現在選択されている一致は、カーソルの位置の次の一致を指します。アンカーの検索がオフになっている場合、最初の現在選択されている一致は、ドキュメントの始めから数えた一番目の一致です。
- *選択範囲内での検索*: オンに切り替えられると、現在の選択範囲がロックされ、検索が選択範囲内に制限されます。それ以外の場合、ドキュメント全体が検索されます。テキストの新しい範囲を選択する前に、現在選択されている検索範囲を検索範囲内オプションをオフにしてロックを解除してください。

テキストの置換え

テキスト文字列を置き換えるには、以下を行います:

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



2. 検索フィールド内で置き換えられる文字列を入力し、フィールドの置換内に新規の文字列を入力します。置き換えられる一致するテキストの数と、現在選択されているマッチは、フィールドの置換えの下に表示されます。また、一致の場所は巣クロークバー上でベージュのマーカーで表示されています。また、一致する箇所はスクロールバー内でベージュのマーカーで表示されています。(詳細に関しては、上の *ドキュメント内でのテキストの検索*[132] を参照してください)。例えば、上のスクリーンショットは、文字列 `type` に対して5つの一致が存在することを示しており、3番目の一致が現在選択されています。

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

以下の点に留意してください:

- ドキュメント全体ではなく、選択範囲内でテキストを置き換えるには、以下を行います: (i) 選択をマークする。(ii) *選択範囲内で検索* 133 オプションをオンに切り替えて、選択をロックする。(iii) 検索と置き換えテキスト文字列を入力します。(iii) 「**次を置き換える**」または「**全てを置き換える**」を必要に応じてクリックします。他の選択範囲内のテキストを置き換えるには、現在の選択範囲を*選択範囲内で検索* 133 オプションをオフに切り替えてロック解除し、新しい選択を指定し、*選択範囲内で検索* 133 オプションをオンにして新しい選択をロックします。
- 置換えのアクションを元に戻す場合は、「**Ctrl+Z**」または、「**編集 | 元に戻す**」を選択します。

正規表現の使用

正規表現 (regex) を使用して、テキスト文字列を検索することができます。これを行うには、*正規表現*のオプションをオンに切り替えます (上の *検索オプション* 133 を参照してください)。これにより、検索用語フィールド内のテキストが正規表現として評価されます。正規表現の構築へのヘルプは、検索用語フィールドの右にある「**正規表現 ビルダー**」ボタンをクリックしてください (*下のスクリーンショット参照*)。ビルダー内のアイテムをクリックして、正規表現メタ文字を対応する検索用語フィールドに入力します。下のスクリーンショットは、電子メールアドレスを検索するための簡単な正規表現を表示しています。メタ文字に関する簡単な説明は、下の*正規表現 メタ文字* 134 セクションを参照してください。

正規表現メタ文字

正規表現 メタ文字のリストは以下の通りです。

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

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

特別文字の表記
以下の式に注意してください。

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

正規表現のサンプル

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

サンプル 1: 要素の名前変更

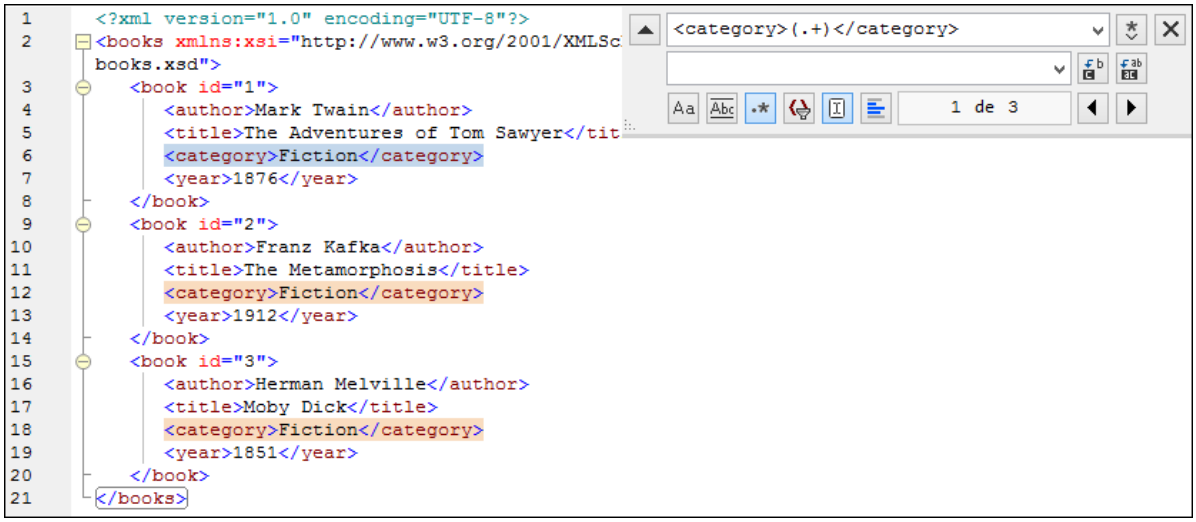
サンプル XML コードリストには、ブックのリストが含まれています。このサンプルの目的を、の各ブックの `<Category>` 要素を `<Genre>` と置き換えることとします。この目的を達成するために正規表現を使用します。

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

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

必要条件を満たすには、以下のステップを踏んでください：

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



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

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

名前を変更されました。

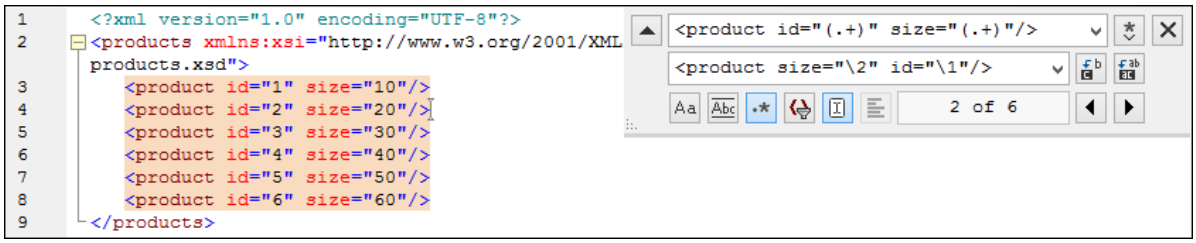
サンプル2: 属性の順序を変更する

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

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

必要条件を満たすには、以下のステップを踏んでください：

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



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

4.2.6 テキストビューでの入力ヘルパー

テキストビューで利用可能な入力ヘルパーの内容は編集中のドキュメントの種類により変化します。標準的なドキュメントの種類に対する入力ヘルパーは以下に示されています。入力ヘルパーの一般的な使用方法については以下に記述されています。138特定のドキュメントの種類に対応したその他の機能については、それぞれのドキュメントの種類のセクションでそれぞれ記述されます。

- *XML*：要素（下のスクリーンショット）、属性、エンティティ

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

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

入力ヘルパーアイテムの表示と使用方法

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

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

メモ 大きなファイルでは、自動補完と入力ヘルパーを無効化し、より速いロードと編集を行うことができます。 しきいファイルのサイズはユーザーにより指定することができます。詳細はオプション | 編集 1021 のセクションを参照してください。

4.2.7 分割ビュー

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

分割ビューと単一ビュー間の切り替え

アクティブなドキュメント の分割ビューを以下のように作成します：

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

```
1   <?xml version="1.0"?>
2   <!DOCTYPE book SYSTEM "book.dtd">
3   <book>
4     <title>Data on the Web</title>
5     <author>Serge Abiteboul</author>
6     <author>Peter Buneman</author>
7     <author>Dan Suciu</author>
8     <section id="intro" difficulty="easy" >
9       <title>Introduction</title>
10      <p>Text ... </p>
11      <section>
12        <title>Audience</title>
13        <p>Text ... </p>
14      </section>
15      <section>
16        <title>Web Data and the Two Cultures</title>
17        <p>Text ... </p>
18        <figure height="400" width="400">
19          <title>Traditional client/server architecture</title>
20          <image source="csarch.gif"/>
21        </figure>
22        <p>Text ... </p>
23      </section>
24    </section>
25    <section id="syntax" difficulty="medium" >
```

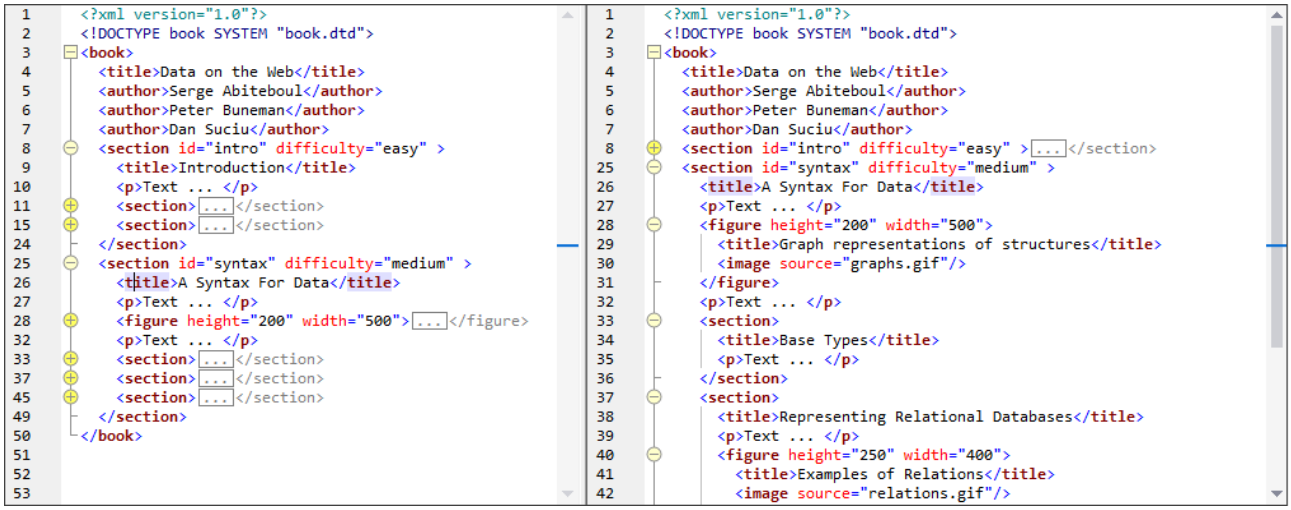
分割ビューから単一ビューに戻るには、以下の1つを行います:

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

メモ　分割ビューは各ドキュメントのために個別に作成されます。

分割ビュー内でのナビゲートを編集

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



以下の点に注意してください：

- 分割ビューの両方のビュー内で使用することのできるすべての機能の表示 124 （ラインの番号づけとソースの折りたたみなど）、編集機能 127 、移動機能 129 など（*上のスクリーンショットを参照*）。
- 分割ビューの各ビュー内で個別にスクロールし、ナビゲートすることができます。
- 各ビュー内で個別にソースの折りたたみを使用することができます。
- 入力ヘルパーアクションを含む、ドキュメント上のすべての編集アクションは両方のビュー内で反映されています。

4.2.8 テキストビューショートカット

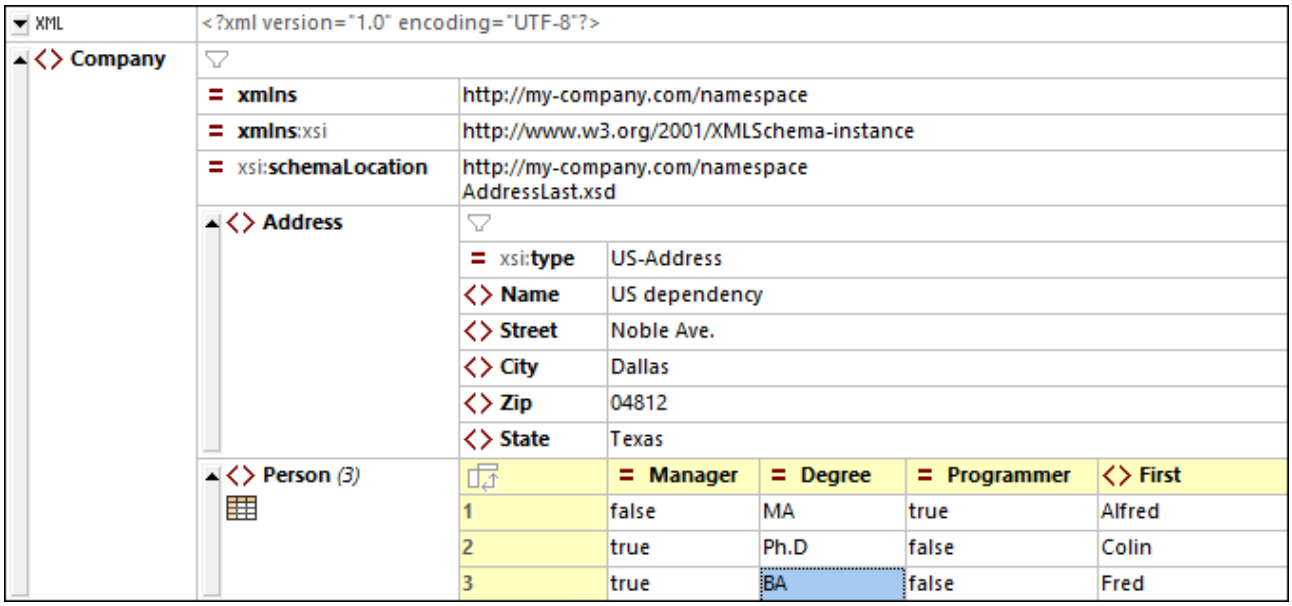
テキストビューコマンドで頻繁に使用されるデフォルトのショートカットは、以下にリストされているとおりです。カスタマイズダイアログのキーボードタブ 1003 内でデフォルトのショートカットを変更することができます。

テキストビューコマンド

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

4.3 グリッドビュー

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



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

- グリッドビュー内のXML 236
- JSON グリッドビュー 475
- DTD 282

グリッドビューのカスタム化

- 列のサイズを調整するには、カーソルを適切な枠線にポイントして希望する幅を達成するためにドラッグします。
- 最も大きいエントリの幅に列をサイズ調整するには、その列の右のグリッドラインをダブルクリックします。
- 全てのコンテンツを表示するために列の幅を調整するにはメニューアイテム 表示 | 最適な幅 968 コマンドを選択、または、グリッドビューツールバー内の最適な幅アイコンをクリックします。
- セルの高さはコンテンツにより決定されます。メニューオプション [ツール | オプション | ビュー | 拡張されたグリッドビュー、[セルの高さを xx 行に制限する] により調節することができます。

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

4.3.1 グリッドビューでの編集

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

要素屋属性名の挿入または追加

選択されたアイテムに関連する要素や属性を挿入するには、XML メニューのコマンドを使用するか、**属性と要素ツールバー**

にあるアイコンを使用します。このツールバーが表示されていない場合、メニューオプションの「**ツール | カスタマイズ**」を選択して、**ツールバー**タブから**属性と要素**にチェックを入れてください。ツールバーにあるこれらアイコンコマンドとXML メニューコマンドについては、XML メニュー[812]の項を参照ください。
要素または属性を挿入または追加するには:

1. 挿入または追加する要素または属性に関連するアイテムを選択します。
2. 属性と要素ツールバー内の適切なアイコンをクリックするか、XML メニュー[812]から適切なコマンド(**挿入**、**追加**、または**子を追加**)を選択します。
3 目的の要素または属性をポップアップから選択し、クリックするか「Enter」を押下します。それ以外にも、目的の要素または属性の名前を入力することができます。

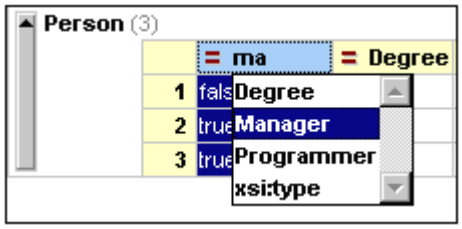
グリッドビューで選択されているものによって、入力ヘルパー[147]の内容が常に変わり続けるのに気がついたかと思います。情報ウィンドウ[1045]に表示される情報も、選択されている要素や属性に従って常に変わり続けます。

グリッドビューにおける改行とタブ

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

要素または属性の名前を編集する

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



要素名または属性名を編集するには:

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

名前空間プレフィックスを追加する

「XML | 名前空間プレフィックス[847]」コマンドによりグリッドビューの選択されたノードとその子孫(要素と属性)に名前空間プレフィックスを追加することができます。名前空間はこれとは別に宣言されなければなりません。

グリッドビューコンテキストメニュー

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

スペルチェック

グリッドビューでは、内蔵の言語辞書を使用することでドキュメントのスペルチェックを行うことができます。ユーザー辞書を作成することで、言語処理に含まれていない単語を加えることができます。詳細については、スペリング[974] および スペルチェッカーのオプション[978] コマンドを参照ください。

XML 検証

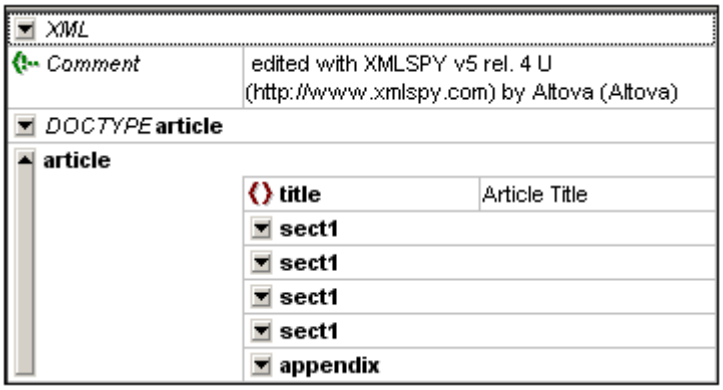
グリッドビューではユーザーの入力を部分的に検証することができます。例えば、属性値における出現とパターンのエラーを検知することが可能です。しかしテキストだけを含めることができる要素に子要素を追加しても、エラーとは検知されません。XML ドキュメントが妥当であることを保証するには、テキストビューへ切り替えた後に検証処理を行なってください。

4.3.2　グリッドビューテーブル

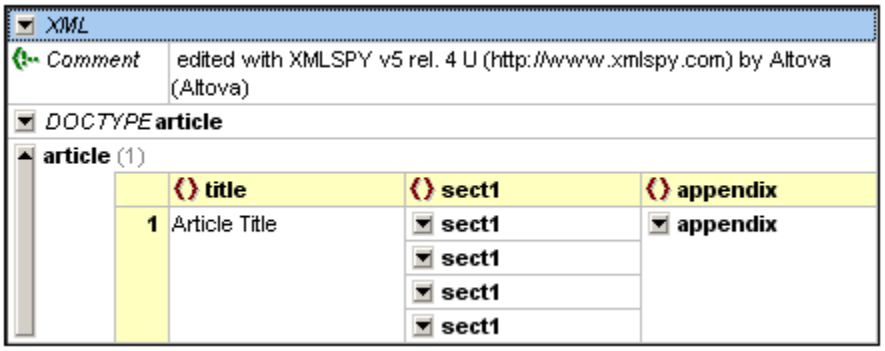
テーブルビューは再帰的な要素をテーブルの形で表示するグリッドビューに統合されたビューです。テーブルビューは通常のグリッドビューとは違い、テーブルとして表示されている要素の子となる型をそれぞれ列として表現します。それから全ての列や選択に対してのプロパティなどを設定することができます。例えば、以下のようなXMLドキュメントを考えてみましょう:

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

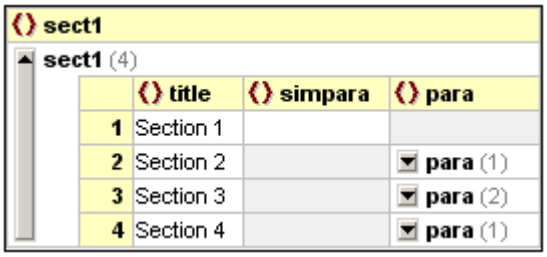
ドキュメントの要素は**article** で、article は以下の子要素を含んでいます: title 要素 1 つ、**sect1** 要素 4 つ、**appendix** 要素を 1 つ。**sect1** と**appendix** 要素はそれぞれ 1 つのtitle 要素を含んでおり、任意の数の**para** か**simpara** 要素がそれに続きます。このドキュメントは通常のグリッドビューでは以下のように表示されます:



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



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



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

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

テーブルデータを操作する

以下のようにしてテーブルのデータを操作することができます:

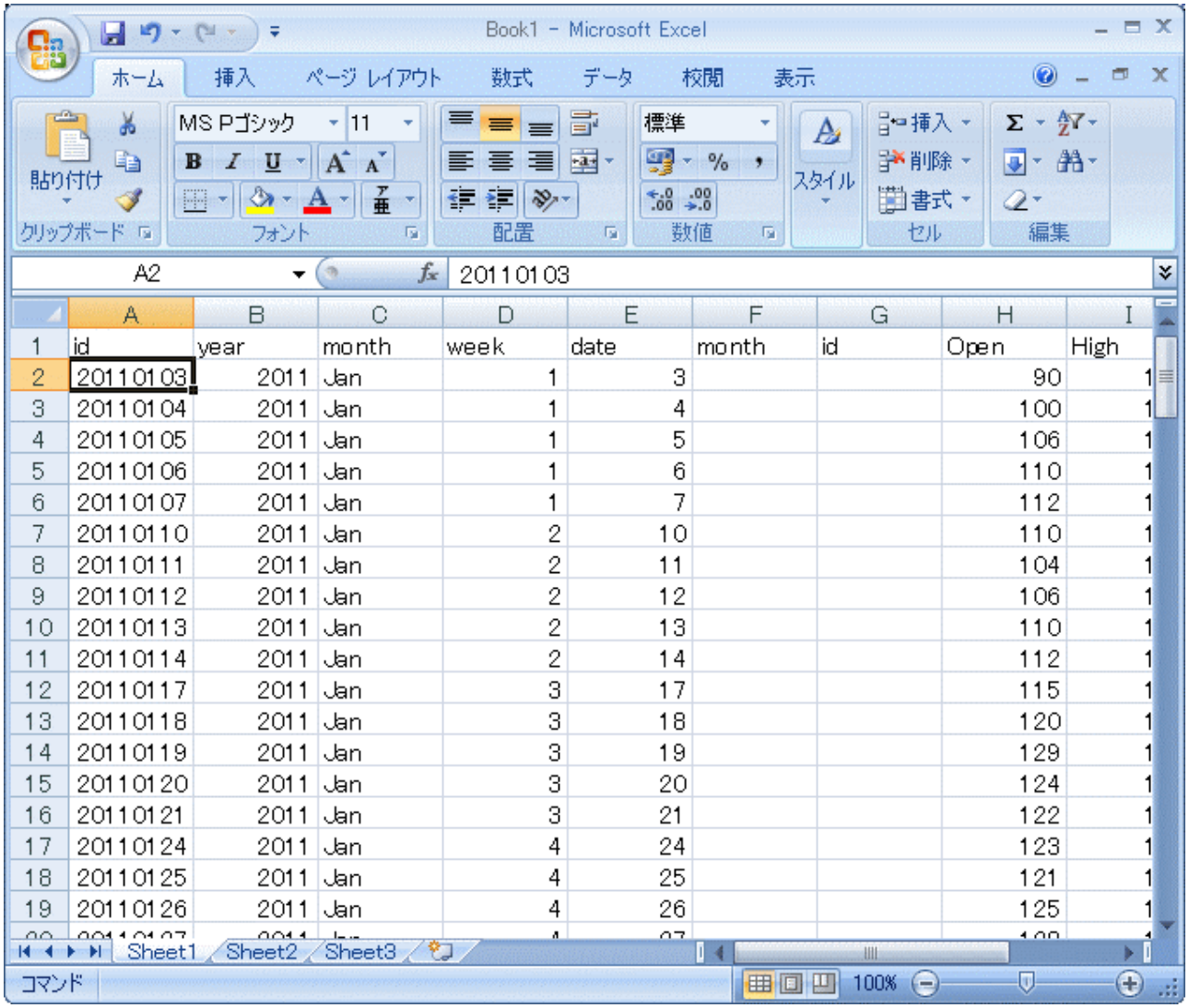
- 列のヘッダをドラッグアンドドロップして列を移動する。
- メニューコマンド「XMI | テーブル | 昇順に並び替え 837」を使用してテキストノードの列のデータをソートする(降順でも可能)。
- メニューコマンド「XML | テーブル | 行を挿入 837」(または**行を追加**)で行の挿入(や追加)を行う。

テーブルビューと外部アプリケーションでデータをやり取りする

テーブルビューによるテーブル構造のデータを利用して、スプレッドシートアプリケーションとテーブルビューとの間でデータのやり取りを行うことができます。テーブルビューから他のアプリケーションにデータを移動したい場合は、テーブル内の必要なノードを選択したあと、構造化テキストとしてコピー 764 オプションを使用して、データを直接(例えば)Excel のシートにコピーすることができます(ノードの選択はセル、列ヘッダ、(下のスクリーンショットの様な)行ヘッダ、またはテーブル全体をクリックすることで行うことができます)。テーブル全体または列ヘッダを選択した場合は、列ヘッダも含めてコピーされます。

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

上のスクリーンショットはテーブルビュー内でテーブルとして表示されている6個の `item` 要素を表示しています。これらの2つの行をExcel シートにコピーするには、構成されたテキストとしてコピーし、Excel シートに張り付けます。結果は以下に示される通りです。詳細に関しては、構造化されたテキストとしてコピーする 764 を参照してください。



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

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

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

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

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

更に複雑なデータのやり取りを行う方法として、XMLSpy にはテキストファイル、W ord ドキュメント、または、データベースを直接 XML データにインポート/エクスポート 938 する機能が搭載されています。

4.3.3 グリッドビューでの入力ヘルパー

要素入力ヘルパー

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

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

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

属性入力ヘルパー

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

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

注意: 既存の属性で複数の出現が許されていないものはグレーで表示されます。

エンティティ入力ヘルパー

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

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

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

注意: 1内部エンティティを追加した場合、その上方をエンティティ入力ヘルパーに表示するためには、ドキュメントを一度保存して再度開く必要があります。

4.3.4 グリッドビューショートカット

グリッドビューコマンドで頻繁に使用されるデフォルトのショートカットは、以下にリストされているとおりです。カスタマイズダイアログのキーボードタブ 1003 内でデフォルトのショートカットを変更することができます。。

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

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

ファイルとアプリケーションコマンド

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

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

その他のキー

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

編集コマンド

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

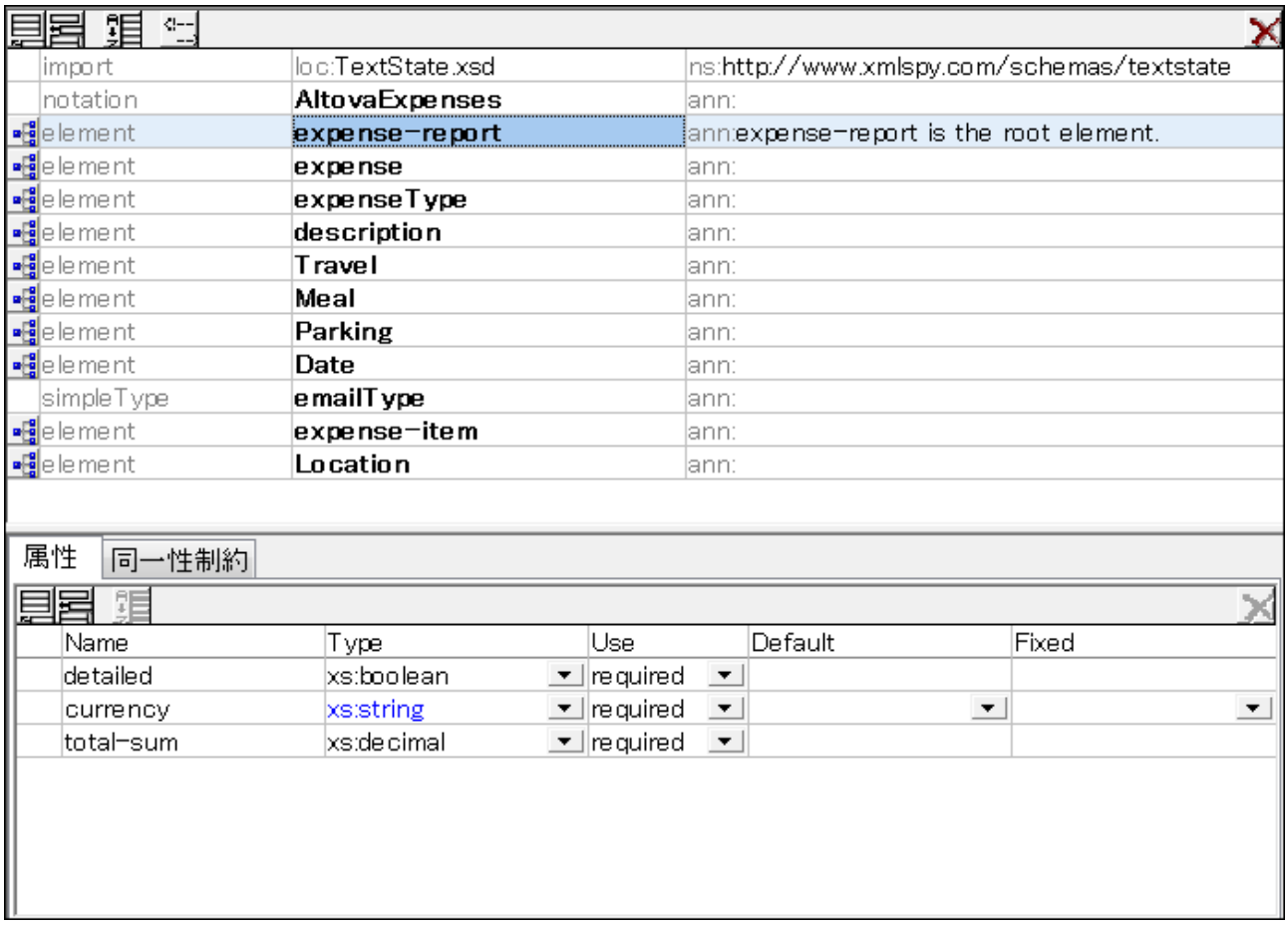
グリッドビューコマンド

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

4.4 スキーマ ビュー

Altova Web サイト: XML スキーマエディター

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



上部ペイン: スキーマのデザイン

スキーマビュー上部のペインでは2 つのモードを切り替えることができます:

- スキーマ概要 156: スキーマの全グローバルコンポーネント（グローバル要素や複合型など）がシンプルなテーブルリストで表示されます（スクリーンショットを参照 156）。グローバルコンポーネントのアイコンをクリックすることで、そのグローバルコンポーネントがコンテンツ モデルビューで表示されます。すべてのグローバルコンポーネントがコンテンツ モデルを持てるわけではない（例えば単純型）ということご注意してください。
- コンテンツ モデルビュー 168: 選択されたグローバルコンポーネントのコンテンツ モデルが表示されます（スクリーンショットを参照 168）。ペイン左上にあるグローバルの表示アイコンをクリックすることで、スキーマ概要が表示されます。

コンテンツ モデル ビューへの切り替え: コンテンツ モデルを含むグローバル コンポーネントに対して利用でき、グローバル コンポーネントのコンテンツ モデルがコンテンツ モデル ビュー[168]にて表示されます。

グローバルの表示: コンテンツ モデル ビューで選択することができ、スキーマ概要[156]への切り替えを行います。

下部ペイン: 属性、アサーション、同一性制約

スキーマ ビューの下部には、(上部ペインの) デザインで選択されているコンポーネントの属性、アサーション、同一性制約[187]の定義が表示されます (スクリーンショットを参照[187])。このペインは便宜上 AAIDC ペインと呼びます。

- XSD 1.0 モードでは、(i) 属性[188]と(ii) 同一性制約[194]という 2 つのタブが下部ペインに表示されます。
- XSD 1.1 モードでは、(i) 属性[188]、(ii) アサーション[191]、(iii) 同一性制約[194]の 3 つのタブが下部ペインに表示されます。

AAIDC ペインはスキーマ概要で常に表示されるほか、コンテンツ モデル ビューにて表示されることもあります。コンテンツ モデル ビューでは、3 種類の定義 (属性、アサーション、同一性制約) すべてを AAIDC ペインではなくデザイン内部に表示することもできます。スキーマ デザイン ツール バーの (i) **属性を図に表示**、(ii) **アサーションを図に表示**、(iii) **制約を図に表示** アイコンを切り替えることで、これらの表示を切り替えることができます。スキーマ表示構成ダイアログ (「スキーマ設計 | 表示設定」[873]) にてこれらの設定を指定することもできます。AAIDC ペインのすべての定義型が図内に表示され、コンテンツ モデル ビューにて下部ペインが表示されなくなります。

スキーマ設定

スキーマ設定ダイアログ (「スキーマ設計 | スキーマ設定」[865]) へはスキーマ ビューからアクセスすることができ、アクティブなスキーマに対するグローバルな設定を定義することができます。これらの設定は `xs:schema` 要素の属性に対応します。

このセクションの構成

このセクションは、以下に示されるサブセクションにより構成されます:

- XSD モード: XSD 1.0 と1.1[153]: 2 つの編集モードから選択を行えます。
- スキーマ概要[156]: グローバル コンポーネントのプロパティを編集します。
- コンテンツ モデル ビュー[168]: 個々のグローバル コンポーネントのコンテンツ モデルを編集します。
- 属性、アサーション、同一性制約[187]: コンポーネントに含まれるこれらプロパティを定義します。
- 入力ヘルパー[201]: コンポーネントに含まれる様々なプロパティを素早く定義します。
- スマート制約[215]: 基底型から派生した型を視覚的に作成ならびに編集します。
- xml の使用: 固定された属性[220]: `base`、`id`、`lang`、`space` 属性をスキーマ コンポーネントへ視覚的に追加します。
- 戻ると進む: 位置の移動[222]: 以前に表示された内容を表示するためのスキーマ ビュー機能について説明します。

SchemaAgent へ接続

XMLSpy からは SchemaAgent への接続を行い、他のスキーマから得られたコンポーネントを GUI に表示して、これらコンポーネントを編集されているスキーマにて使用することができます。XMLSpy にて SchemaAgent を使用する方法については、「SchemaAgent を使用する」[289]のセクションを参照ください。

スキーマの中を検索

スキーマの中を検索機能により、スキーマ内部のインテリジェントな検索 (スキーマに関連した条件による絞り込み) を行うことができます。例えば、検索の結果を特定種類のコンポーネントへ絞り込むことで、より効率的な検索を行うことができます。スキーマの中を検索機能の詳細については、「DTD とXML スキーマ」[307]のセクションを参照ください。

(C) 2015-2021 Altova GmbH

4.4.1 XSD モード: XSD 1.0 と 1.1

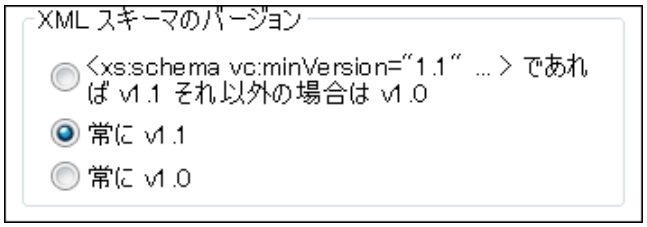
スキーマビューでは、 XML スキーマ(XSD) が編集されるか、そして、XML Schema 1.0 仕様(XSD 1.0) または XML Schema 1.1 仕様(XSD 1.1) に従い検証されるかを選択することができます。**XSD モード** は、次の2 つの設定をベースにした編集で使用されます: アプリケーション内、他のXSD ドキュメント内。

XSD モードの選択

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

アプリケーション全体モード

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



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

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

ドキュメント固有のモード

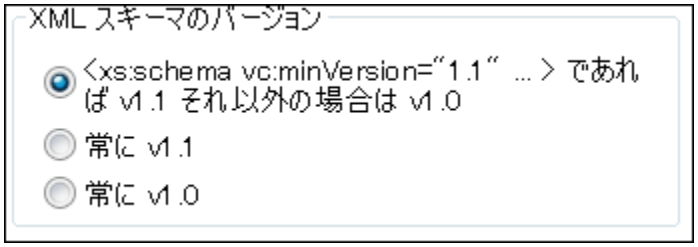
XSD ドキュメント内にXSD モード 情報を保存することを選択できます。ドキュメントがロードされると、自動的にスキーマビューのXSD モードへの切り替えを有効化します。スキーマデザイン ツール バー内の **XSD 1.0** または**XSD 1.1** ボタンをクリックすることにより、XSD モード 情報をXSD ドキュメントに追加することができます。(*下のスクリーンショット*). これを行うにあたり、選択されたモードは、 トップレベル `xs:schema` 要素の `vc:minVersion` 属性内に保存されます (`vc:minVersion` の値は、手動でテキストビューに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

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

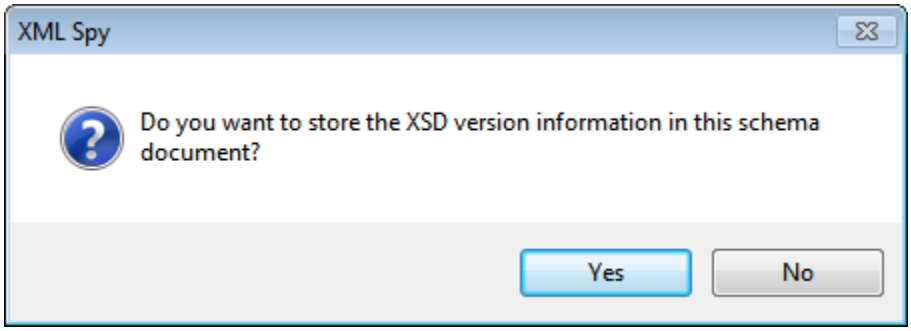
クリーンショット)。ドキュメント内の他のノード名がこの名前空間に存在しない場合、**XSD 1.0** ツールバーボタンをクリックすると、名前空間宣言が削除されます。

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

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



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



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

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

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

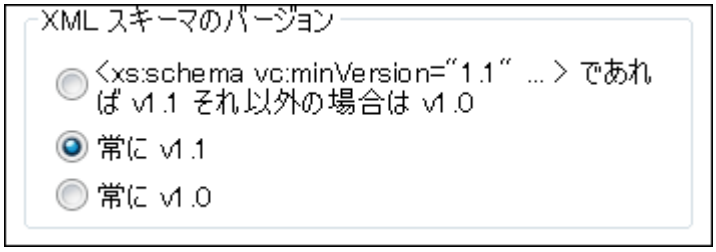
新しいドキュメントのXSD モード

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

 (C) 2015-2021 Altova GmbH

有効化されたXSD モード

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



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

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

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

XSD モード 機能

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

XSD 1.0 モードが有効化される場合:

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

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

XSD 1.0 モード内でのXSD 1.1 機能の扱いかた

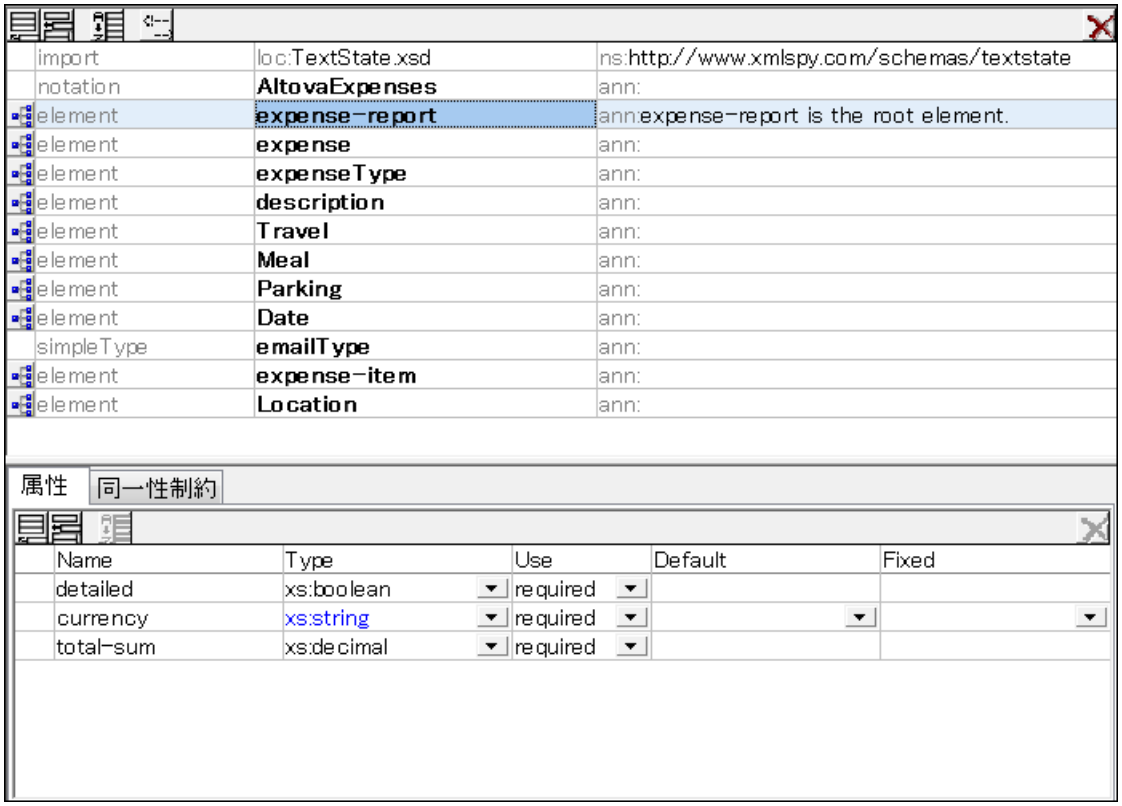
XSD 1.0 によりサポートされないXSD 1.1 機能 (例えば アサーション) がドキュメント内に存在する場合、その 機能が表示され、 そして以下に説明されるXSD 1.0 モードにより取り扱われます。

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

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

4.4.2 スキーマ概要

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



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

属性　同一性制約

Name	Type	Use	Default	Fixed
detailed	xs:boolean	required		
currency	xs:string	required		
total-sum	xs:decimal	required		

グローバルコンポーネントの挿入や追加、削除のほか、プロパティを修正することができます。プロパティを修正するには、スキーマ概要リストにてグローバルコンポーネントを選択してください。
選択されたグローバルコンポーネントのプロパティは、詳細入力ヘルパー[205]、ファセット入力ヘルパー[207]、または属性/アサーション/同一性制約（AAIDC）ペイン[187]で編集することができます。

コンテンツモデルを含んでいるグローバルコンポーネントには、グローバルコンポーネントリストの左に「**コンテンツモデルビューへの切り替え**」アイコンが表示されます。このアイコンをクリックすることで、そのコンポーネントがコンテンツモデルビュー[168]で表示され、コンポーネントのコンテンツモデルを編集することができます。

コンテンツモデルビューへの切り替え: コンテンツモデルが含まれるグローバルコンポーネントにて表示され、コンポーネントのコンテンツモデルをコンテンツモデルビュー[168]にて表示します。

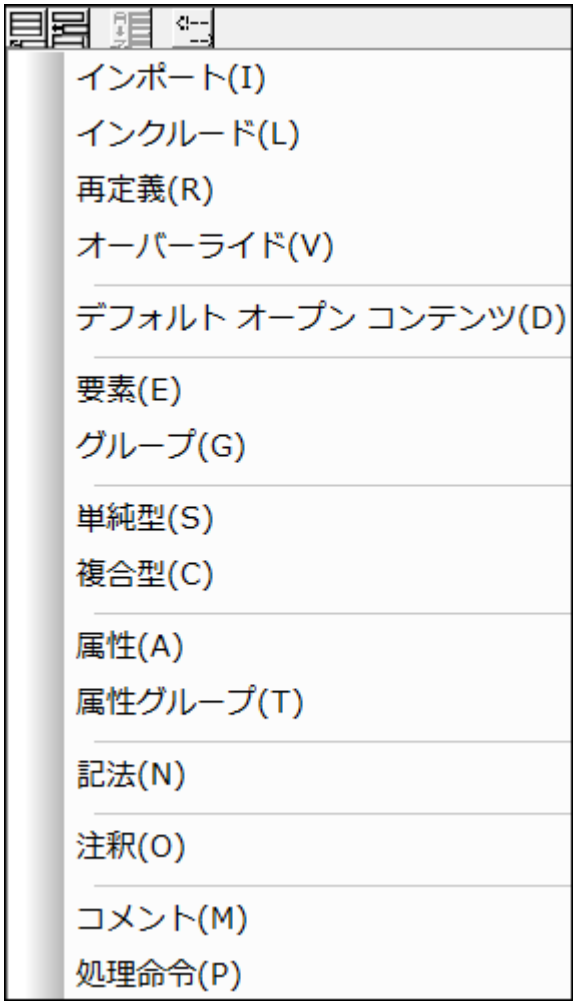
グローバルの表示: コンテンツモデルビューにて表示され、クリックすることでスキーマ概要[156]が表示されます。

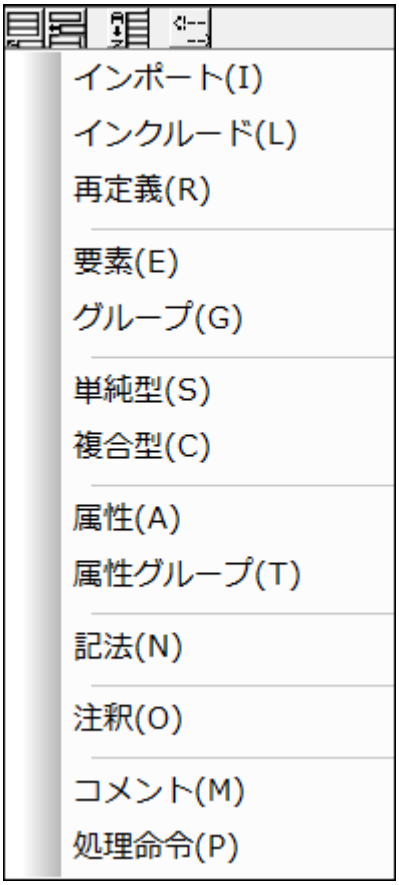
このセクションでは、最初にスキーマ概要のGUI メカニズム[157]が説明され、各種グローバルコンポーネント[161]の特徴について記述されます。

4.4.2.1 GUI メカニズム

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

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





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

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

スキーマ概要での編集

スキーマ概要の編集機能に関する以下の点に留意してください：

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

スキーマ概要と関連するアイコン

	グローバルコンポーネントの追加: グローバルコンポーネントのリストの最後に新たなグローバルコンポーネントを追加します。定義によりコンポーネントがドキュメントの最初に出現する必要のある場合、リストの最初に追加されます。
	グローバルコンポーネントの挿入: 選択されているコンポーネントの直前にグローバルコンポーネントを追加します。定義によりコンポーネントがドキュメントの最初に出現する必要のある場合、リストの最初に追加されます。
	並べ替え: コンポーネントの並び替えダイアログが表示され、実際に並び替えを行う前に、並び替えを行うための条件（種類の前に名前で並び替えを行う、またはその逆）を指定することができます。*以下の記述を参照ください。*
	コメント: グローバルコメントの表示を複数行で行うのか、または 1 行で行うのかを指定することができます。*以下の記述を参照ください。*
	コンテンツモデルビューへの切り替え: コンテンツモデルを持つグローバルコンポーネントのために使用することができます。コンテンツモデルビュー 168 内のグローバルコンポーネントのコンテンツモデルを開きます。
	グローバルの表示: コンテンツモデルビュー内で使用が可能です: スキーマ概要 156 に切り替えます。

スキーマ概要とコンテンツ モデル ビューの切り替え

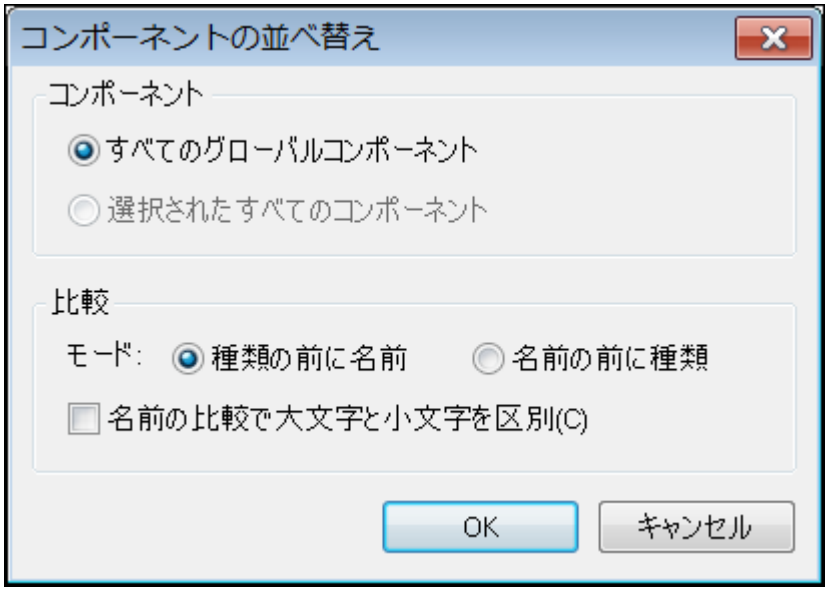
(複合型、要素宣言、モデル グループなどの) グローバル コンポーネントには、コンポーネント名の左側に「**コンテンツ モデル ビューへの切り替え**」アイコンが表示されます (*上記にあるアイコンのリストを参照ください*)。これにより、コンポーネントの構造とコンテンツが記述されるコンテンツ モデルを、これらグローバル コンポーネントに含めることができることが示されます。

このアイコンをクリックすることで、スキーマ概要から、そのグローバル コンポーネントのコンテンツ モデル ビュー168 への切り替えが行われます。その他のグローバル コンポーネント (注釈や単純型、属性グループなど) にコンテンツ モデルは含まれておらず、「**コンテンツ モデル ビューへの切り替え**」アイコンも表示されません。**グローバルの表示** アイコンをクリックすることでコンテンツモデルビューからスキーマ概要に切り替えることができます (*上のアイコンのリストを確認してください*)。

グローバル コンポーネントの並べ替え

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

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



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

並べ替えの順序はスキーマのテキストに反映されます。

グローバル コメント: 行の表示モード

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

(C) 2015-2021 Altova GmbH

element	Desc
comment	This is the first line of this comment. This is the second line. This is the third line.
element	para

element	Desc
comment	This is the first line of this comment.
element	para

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

4.4.2.2 グローバル コンポーネント

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

スキーマ概要内でのグローバル コンポーネントの作成

グローバルコンポーネントは通常スキーマ概要[156]で作成ならびに編集されます。スキーマ概要では**追加**[157] または[157] **挿入**[157] アイコン[157]を介して追加されます。（グローバルコンポーネントがコンテンツモデルを有している場合 *下のテーブルを参照*）グローバルコンポーネントのコンテンツ モデルは、そのグローバルコンポーネントのコンテンツ モデルビュー[168]にて作成ならびに編集することができます。（コンポーネント名の左側にある**コンテンツ モデル ビューへの切り替え**アイコンをクリックすることで、コンテンツ モデルビューが表示されます[168]）。

スキーマ概要にて作成されたグローバルコンポーネントによってはコンポーネント入力ヘルパー[201]へも追加されます。コンポーネントにコンテンツ モデルが含まれる場合、コンポーネント入力ヘルパーでその名前をダブルクリックすることでコンポーネントがコンテンツ モデルビュー[168]で開かれ、編集を開始することができます。

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

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

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

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

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

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

インクルード、インポート、再定義、オーバーライド

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

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

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

現在編集中のスキーマドキュメントにてスキーマが(インクルード、インポート、再利用、またはオーバーライドにより) 再利用されると、そのグローバルコンポーネント、名前空間、同一性制約が編集中ドキュメントのコンポーネント入力ヘルパー 201に表示されます。

再定義

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

(C) 2015-2021 Altova GmbH

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

コンポーネントを再定義するには、以下を行います:

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

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

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

オーバーライド

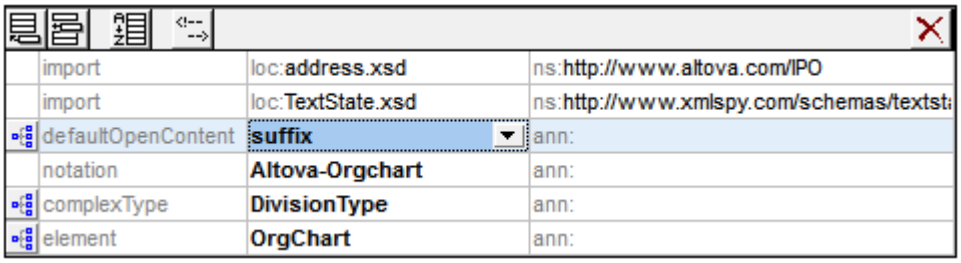
`override` 要素では、以下のコンポーネントを定義することができます: 複合型、単純型、グローバル要素、グローバル属性、モデルグループ、属性グループ、記法。`override` 要素内でこれらコンポーネントのうちの1つが定義されている場合、オーバーライドされるすべてのスキーマにある同じ型と名前を持ったすべてのコンポーネントが、このコンポーネントによりオーバーライドされます。オーバーライドされたスキーマは、オーバーライドの `loc` フィールドにて指定されます。

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

デフォルト オープン コンテンツ

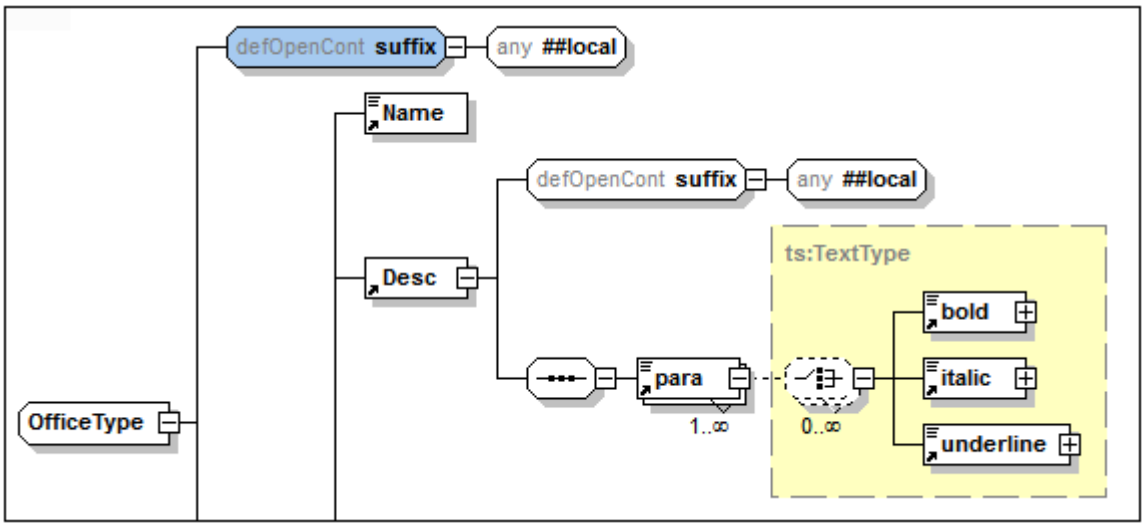
`defaultOpenContent` 要素はXSD 1.1 に新たに追加された仕様で、混合または要素だけのコンテンツを含めることのできる複合型に追加することのできる1つ以上の要素を指定します。XSD 1.1 にて新たに追加された `openContent` 185 要素に似ており、`openContent` 要素が単一の複合型に対して適用されるのに対して、`defaultOpenContent` 要素はスキーマ内にあるすべての複合型に対して適用されます。

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

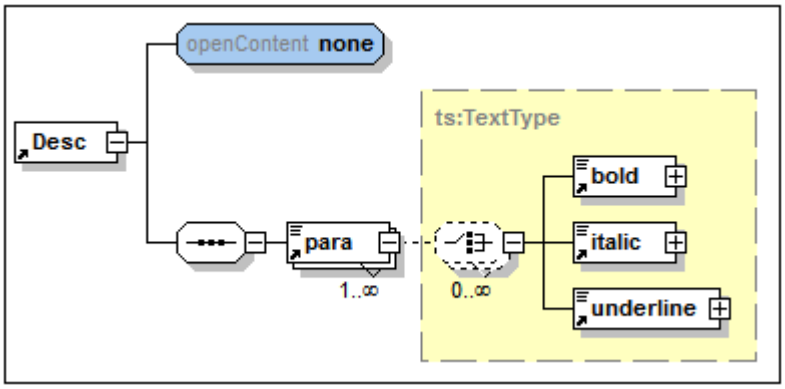


import	loc:address.xsd	ns:http://www.altova.com/IPO
import	loc:TextState.xsd	ns:http://www.xmlspy.com/schemas/textst
defaultOpenContent	suffix	ann:
notation	Altova-Orgchart	ann:
complexType	DivisionType	ann:
element	OrgChart	ann:

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



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



グローバル要素 (element)

スキーマ概要ではグローバル要素を作成することができます。グローバル要素にコンテンツ モデルが含まれる場合、グローバル要素のコンテンツ モデルビューにて定義することができます。要素が選択されている状態で、属性 187 、アサーション 191 、そして同一性制約 194 を AAIDC

ペイン[187]の対応するタブにて定義することができます。要素のファセット[207]ならびに詳細[205]入力ヘルパーにて定義することができます。グローバル要素は複合型から参照することができます。グローバル要素は複合型から参照することができます。

モデル グループ（group）

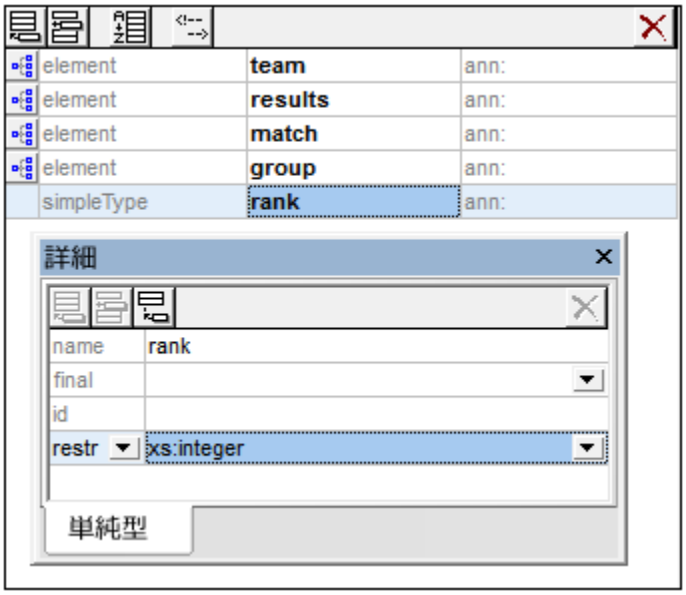
スキーマ概要では名前付きのモデル グループを作成し、複合型にて参照することができます。名前付きのモデル グループ（`xs:group` 要素）により、再利用可能なコンテンツ モデルを予め定義することができます。`sequence` グループ、`choice` グループ、または `all` グループのうちどれかを含めることができます。

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

名前付き単純型（simpleType）

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

詳細入力ヘルパーでは、単純型のコンテンツ（`restriction`、`list`、`union`）と対応する種類（それぞれ基底型、アイテム型、メンバー型）を指定することができます。例えば以下のスクリーンショットでは、単純型の制約の基底型に `xs:integer` が指定されています。詳細については「詳細入力ヘルパー」[205]のセクションを参照ください。ファセットにより単純型の制約を行うには、ファセット入力ヘルパー[207]にあるオプションを使用してください。



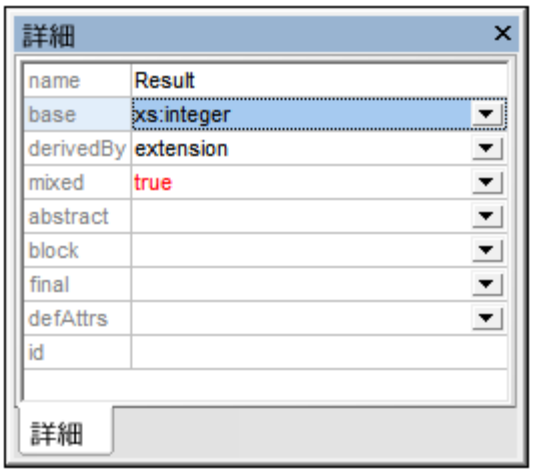
メモ　匿名型は、スキーマ概要[156]または コンテンツ モデル ビュー[168]にて、単純型コンテンツの属性または要素で宣言することができます。（詳細入力ヘルパーにある）`derivedBy` プロパティの値を `restriction`、`list`、または `union` に指定することで、その要素または属性宣言内部に匿名の単純型が作成されます。制約ファセットをファセット入力ヘルパーで、その他のプロパティを詳細入力ヘルパーで定義することができます。

名前付き複合型（complexType）

スキーマ概要では名前付きの複合型を作成し、要素宣言から参照することができます。名前付きの複合型が選択されている状態で、AAIDC ペイン[187]の対応するタブにて属性[187]とアサーション[191]を定義することができます。

複合型には4 種類のコンテンツを含めることができます（*以下のリストを参照*）。詳細入力ヘルパーにて各種コンテンツを指定し、必要に応じてコンテンツ モデル ビューにてコンテンツ モデル[168]を定義することができます。

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



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

メモ 属性とアサーションは（AAID C ペイン）187 にて 4 種類のコンテンツすべてに対してセットすることができます。

メモ ある要素に対するコンテンツ モデルをコンテンツ モデル ビュー 168 で作成することにより、その要素内部で匿名の複合型が作成されます。

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

グローバル属性と属性グループはスキーマ概要にて追加されます。

- **グローバル要素**のプロパティは属性の詳細入力ヘルパーにて定義されます。
- **グローバル属性グループ**を作成すると、グループに属性を以下のように追加することができます: (i) グローバルコンポーネントのリストにてグローバル属性グループを選択する。(ii) AAID C ペイン 187 の属性タブにて属性を追加する。(iii) 各属性のプロパティを選択された属性の詳細入力ヘルパーにて定義することで、属性をグループへ追加する。

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

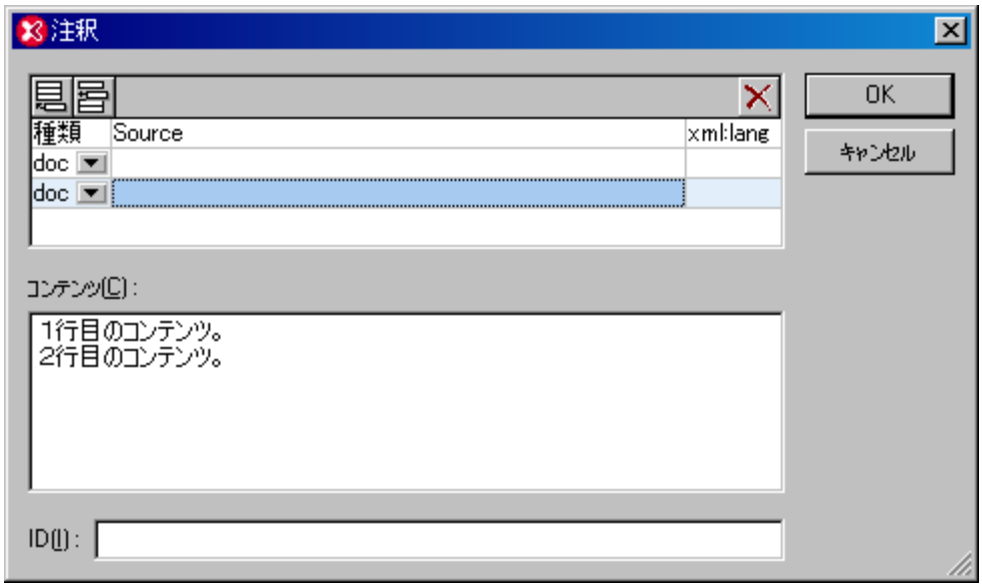
表記（notation）

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

(C) 2015-2021 Altova GmbH

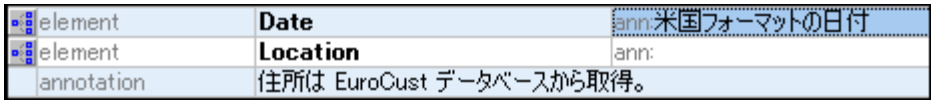
グローバル注釈

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



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

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



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

コメントと処理命令

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

4.4.3 コンテンツ モデル ビュー

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

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

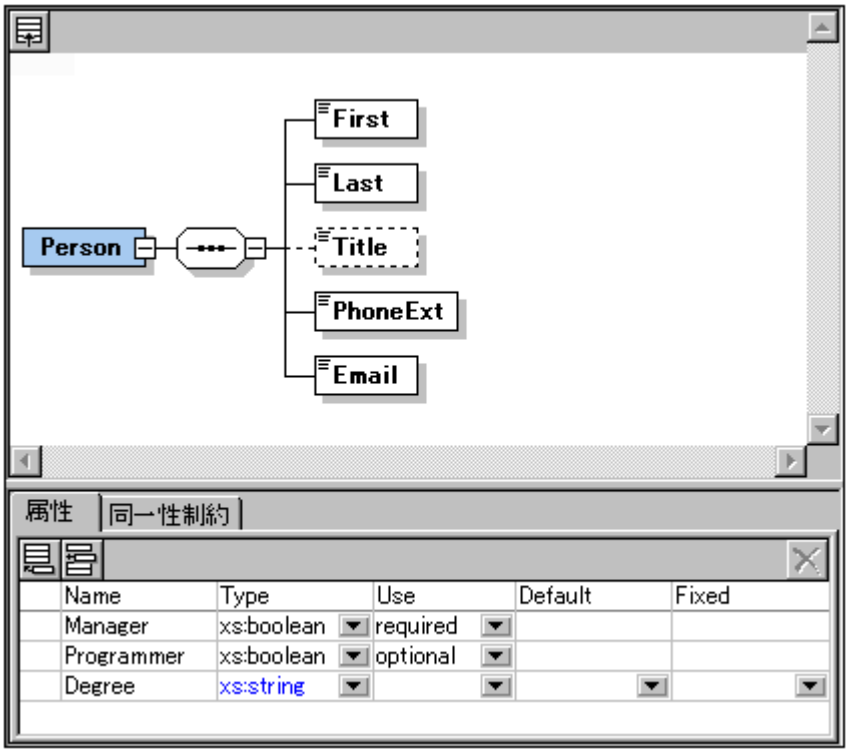
スキーマ概要[156]に示されたこれらのコンポーネントの左には、「**コンテンツ モデル ビューへの切り替え**」アイコンが表示されます。

	コンテンツ モデル ビューへの切り替え: コンテンツ モデルが含まれるグローバル コンポーネントにて表示され、コンポーネントのコンテンツ モデルをコンテンツ モデルビュー[168]にて表示します。
	グローバルの表示: コンテンツ モデルビューにて表示され、クリックすることでスキーマ概要[156]が表示されます。

コンテンツ モデル ビューへの切り替えアイコンをクリックすることで、そのコンポーネントの**コンテンツ モデル ビュー**が表示されます（以下のスクリーンショットを参照）。その他にも、スキーマ概要[156]にて（i）コンポーネントを選択し、メニューから「**スキーマ設計 | 図の表示**」オプションを選択するか、（ii）コンポーネンツ入力ヘルパー[201].にあるコンポーネント名をダブルクリックすることでもコンテンツ モデルビューが表示されます。同時に開くことのできるコンテンツ モデルビューは、スキーマごとに 1 つしかないという点に留意してください。コンテンツ モデルが表示されている場合、「Ctrl」キーを押下しながら現在表示されているコンテンツ モデル内にあるコンポーネントをダブルクリックすることで、そのコンポーネントのコンテンツ モデルへジャンプすることができます。

コンテンツ モデル ビュー全般の説明

コンテンツ モデルビューではコンテンツ モデルがツリーとして表示されます（以下のスクリーンショットを参照）。メニューの「スキーマ設計 | 表示設定[873]」により表示されるスキーマ表示構成ダイアログでは、ツリーの外観を設定することができます。



以下の点に留意してください：

- コンテンツ モデル ツリーにはコンポジッターとコンポーネントの2 種類あります。更に、属性、アサーション、同一性制約、オープン コンテンツをコンポーネントに接続されたボックス内に表示することもできます。
- ツリーの各階層は、コンポジッターにより接続されます。コンテンツ モデルに含めることのできる階層の数に制限はありません。
- オブジェクトを右クリックすることで表示されるコンテキスト メニューから、そのオブジェクトに対して相対的に他のオブジェクトを追加することができます。
- コンテンツ モデル内にあるコンポーネントは、ローカル コンポーネントとして定義またはグローバル コンポーネントを参照することができます。
- ドラッグ アンド ドロップ機能によりオブジェクトを移動することができます。
- キーボード ショートカットにより、オブジェクトをコピー（「Ctrl」 + C）または貼り付ける（「Ctrl」 + V）ことができます。
- オブジェクトのプロパティは詳細入力ヘルパーならびにAAIDC ペイン187にて編集することができます。
- コンポーネントの属性、アサーション、同一性制約はコンテンツ モデル ビュー下部にあるAAIDC ペイン187に表示されます。属性と同一性制約はAAIDC ペイン187の代わりにコンテンツ モデルの図に表示することもできます。この表示オプションはスキーマ表示構成ダイアログ873にてセットすることができます。表示オプションはスキーマ デザイン ツール バーにある「**図に表示**」ボタンからもセットすることができます（以下のスクリーンショットを参照）。

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

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

「グローバルの表示」アイコンをクリックするか、メニュー オプションの「スキーマ設計 | すべてのグローバルを表示」を選択することで、スキーマ概要[156]が表示されます。

4.4.3.1 コンテンツ モデル オブジェクト

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

- コンポジッター[170]: (i) シーケンス、(i) 選択、(iii) すべて
- コンポーネント[171]: (i) 要素、(i) 複合型、(iii) モデル グループ、(iv) ワイルドカード
- その他[173]: (i) 属性、(ii) 属性グループ、(iii) アサーション、(iv) 制約、(v) オープン コンテンツ

これらのオブジェクトの外観をそれぞれ以下に示します。

コンポジッター

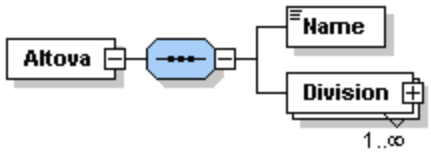
コンポジッターにより、子要素が出現する順序が定義されます。3 つのオプションから選択することができます: シーケンス、選択、および すべて。

以下の操作によりコンポジッターを挿入することができます:

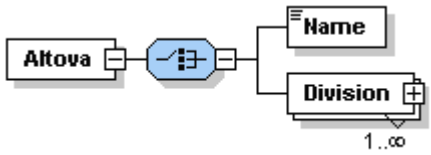
1. 子要素を追加する要素を右クリックします。
2. コンテキスト メニューから「子の追加 | シーケンス (選択 、すべて)」を選択します。

それぞれのコンポジッターは以下のような外観を持ちます。

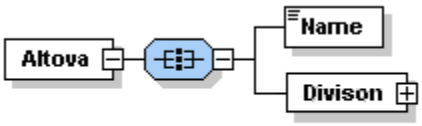
- シーケンス



- 選択



- すべて



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

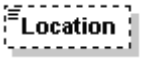
コンポネート

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

- 必須の要素 1 つ

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

- 省略可能な要素 1 つ



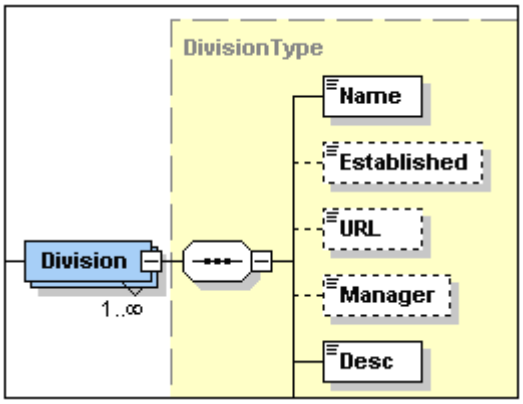
属性、*属性グループ*長方形によりそれが要素であることが示され、点線の境界線によりその要素が省略可能であることが示されます。番号の範囲が示されていないことから、その要素の出現回数が1 回である(`minOcc=0` and `maxOcc=1`) ことが理解できます。上の例では、`Location` が要素の名前となります。
メモ コンテキスト メニューにある「**省略可能**」をクリックすることで、その要素がオプションとなります。

- 複数の必須要素

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

- 子要素を含む複数の必須要素

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

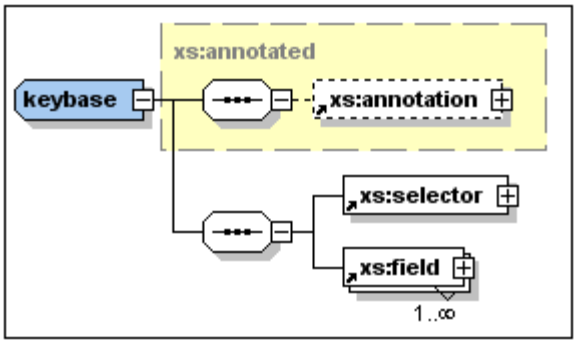


- グローバル要素を参照している要素

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

- 複合型

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



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

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

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

- モデルグループ

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

- ワイルドカード

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

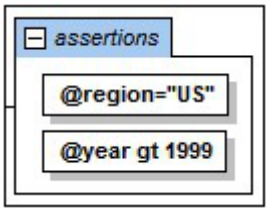
その他のオブジェクト

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

- 属性、属性グループ

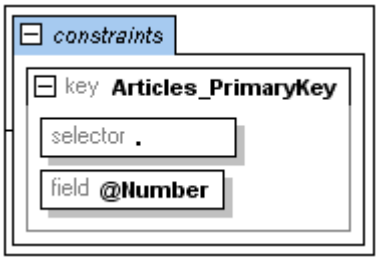
属性、*属性グループ*イタリック体の *'attributes'* という文字と、展開可能な長方形により表示されます。各属性を表す長方形は、(i) 属性が省略可能の場合は破線の境界線、(ii) 属性が必須の場合は実線の境界線で表示されます。属性グループと属性ワイルドカードも *'attributes'* の長方形内に表示されます。
メモ 属性は図の内部と詳細入力ヘルパーの両方で編集することができます。属性は、コンテンツ モデルの図内部またはコンテンツ モデルビュー下部にあるAAIDC ペイン 187内部に表示されます。これら 2 つの表示方法は属性の表示 アイコンをクリックすることで切り替えることができます。要素内にある属性の順序を変更するには、属性をドラッグして、目的の場所で矢印が表示されている状態でドロップしてください。

- アサーション メッセージ コントロール



属性、*属性グループ*イタリック体の *'assertions'* という文字と、展開可能な長方形により表示されます。各アサーションは、ボックス内の長方形として示されます。
メモ アサーションは図の内部と詳細入力ヘルパーの両方で編集することができます。アサーションは、コンテンツ モデルの図内部またはコンテンツ モデルビュー下部にあるAAIDC ペイン 187 内部に表示されます。これら 2 つの表示方法はアサーションの表示 アイコンをクリックすることで切り替えることができます。要素内にあるアサーションの順序を変更するには、アサーションをドラッグして、目的の場所で矢印が表示されている状態でドロップしてください。

- 同一性制約

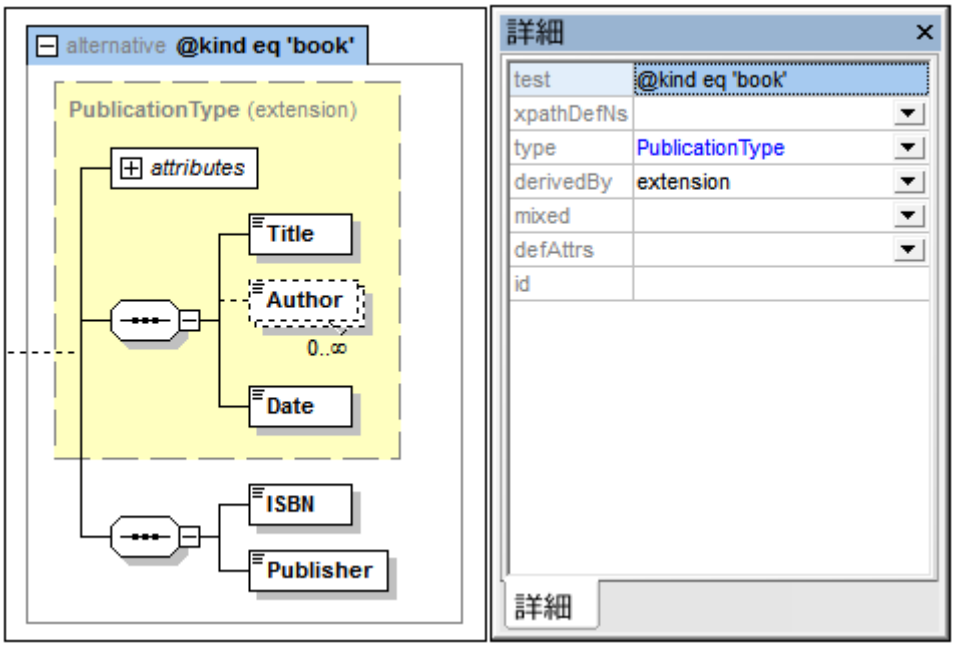


属性、*属性グループ*イタリック体の *'constraints'* という文字と、展開可能な長方形により表示されます。
メモ コンポーネントのコンテンツ モデルに表示された同一性制約により、`key` と `keyref`、そして `unique` 要素により定義された制約が表示されます。`ID` データ型により定義された同一性制約はコンテンツ モデルの図ではなく詳細入力ヘルパーに表示されます。同一性制約はコンテンツ モデルビュー、またはスキーマ概要の同一性制約タブにて表示ならびに編集することができます。コンテンツ モデルビューでは、制約の表示 アイコンにより、制約ボックスの表示を切り替えることができます。

- 条件型の割り当て

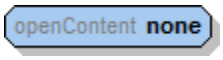
属性、*属性グループ* `alternative` という名前のついた長方形には、評価される XPath 条件式が含まれます（*上のスクリーンショットを参照*）。`alternative` 要素の種類は詳細入力ヘルパーにより指定されます。複合型の場合、展開された `alternative` 要素のボックスに表示され、編集を行うことができます（*以下のスクリーンショットを参照*）。単純型は図に表示されませんが、詳細入力ヘルパーの単純型タブにて定義することができます。

(C) 2015-2021 Altova GmbH



メモ `alternative` 要素はXSD 1.1 にて新たに導入されました。XPath 条件式の結果がtrue となる場合、`alternative` 要素により指定された型が選択された型となります。`alternative` 兄弟の中で、最初にtrue と評価された `alternative` 要素が選択されるため、alternative 要素の順序には意味があります。`alternative` 要素のボックスをドラッグすることで、順序は変更することができます。詳細は条件型アサーション[181]を参照してください。

- デフォルト オープン コンテンツ、オープン コンテンツ



属性、*属性グループ* `defaultOpenContent` と `openContent` 要素は、`openContent` と `defOpenContent` というラベルとともにコンテンツ モデルビューに示されます。ワイルドカード要素コンテンツは `any` ボックスにより示されます（*上のスクリーンショットを参照*）。
メモ `defaultOpenContent` と `openContent` 要素はXSD 1.1 にて新たに追加されたものです。デフォルト オープン コンテンツはグローバル コンポーネントで、スキーマ概要[156]にて作成されます。特定コンポーネントのコンテンツ モデルを表しているコンテンツ モデルビューでは、デフォルト オープン コンテンツをそのコンポーネントで使用されるオープン コンテンツにより置き換え、スキーマのデフォルト オープン コンテンツをオーバーライドすることができます。コンポーネントの子としてオープン コンテンツを追加してください。デフォルト オープン コンテンツのボックスがオープン コンテンツのボックスに置き換えられます。コンテンツ モデルビューでは、オープン コンテンツの `mode` 属性と、そのワイルドカード要素の名前空間を図のなかと詳細入力ヘルパーの両方で編集することができます。任意の複合型のコンテンツ モデルビューから、スキーマ全体に対して適用されるデフォルト オープン コンテンツを修正することもできます。

メモ

- スキーマ表示構成ダイアログ[873]にて指定された定義済みの詳細情報の表示は、定義済みの詳細情報ツール バー アイコンにより切り替えることができます。

- 属性、アサーション、同一性制約の表示アイコンをクリックすることにより、これらの情報をコンテンツ モデルの図の中で表示するか、または（コンテンツ モデルビュー下部にある）AAIDC ペイン[187] 内部に表示するかを切り替えることができます。
- コンテンツ モデルビューでは、「Ctrl」 キーを押下しながら図の中に表示されているグローバルコンポーネントの名前をダブルクリックすることで、そのグローバルコンポーネントのコンテンツ モデルへジャンプすることができます。
- 基底型の名前をダブルクリックすることでも、基底型のコンテンツ モデルへ移動することができます。

4.4.3.2 コンテンツ モデル ビューでの編集

コンテンツ モデルビューで編集を行う方法は、以下のカテゴリーにより説明されます:

- コンテンツ モデルビューの構成[176]
- 属性、アサーション、同一性制約[176]
- コンテンツ モデルビュー アイコン[176]
- コンテキスト メニューからの操作[177]
- キーボード ショートカットとドラッグ アンド ドロップ[179]
- コンポーネント プロパティ[180]
- 注釈[180]
- コメントと処理命令[181]
- コンテンツ モデルのドキュメント化[181]

コンテンツ モデルビューの構成

スキーマ表示構成ダイアログ「**スキーマ設計 | 表示設定**」では、スキーマ全体に対するコンテンツ モデルビューの構成を指定することができます。構成のオプションに関する詳細については、ユーザーリファレンスにある「構成ビュー」[873] セクションを参照ください。ここで行われる設定はスキーマ全体、そしてスキーマドキュメンテーションの出力と印刷の出力にも適用されることに注意してください。

属性、アサーション、同一性制約

コンポーネントの属性、アサーション、そして同一性制約は、コンテンツ モデルビューの下部に表示されるAAIDC ペイン[187]に、またはコンテンツ モデルビューそのもの、つまり図の中に直接表示されます。後者の表示オプションは、スキーマ表示構成ダイアログ[873]で設定することができます。またスキーマ設計ツールバーにある**図の中に表示**ツールバーボタンからセットすることができます（*下のスクリーンショット、およびアイコンのリストを参照してください*）。

属性、アサーション、同一性制約の挿入ならびに編集方法については、「属性、アサーション、同一性制約」[187] のセクションを参照ください。

コンテンツ モデルビュー アイコン

	グローバルの表示: コンテンツモデルビュー内で使用が可能です。スキーマ概要[156] に切り替えます。
	定義済みの詳細の追加: コンテンツ モデルビューが表示されている際にスキーマ設計ツールバーで有効になります。コンポーネント内にある定義済みの詳細情報の表示を切り替えます。
	属性を図に表示: コンテンツ モデルビューが表示されている際にスキーマ設計ツールバーで有効になります。（オンに切り替え）属性の表示を図または属性タブで行うか指定します。

(C) 2015-2021 Altova GmbH

	アサーションを図に表示: コンテンツ モデルビューが表示されている際にスキーマ設計ツール バーで有効になります。(オンに切り替え) アサーションの表示を図またはアサーション タブで行うか指定します。
	制約を図に表示: コンテンツ モデルビューが表示されている際にスキーマ設計ツール バーで有効になります。(オンに切り替え) 同一性制約の表示を図または同一性制約タブで行うか指定します。
	同一性制約の可視化: コンテンツ モデルビューが表示されている際にスキーマ設計ツール バーで有効になります。同一性制約の情報表示を切り替えます。

コンテキスト メニューからの操作

コンテンツ モデルビューにおける編集のいくつかは、コンテンツ モデルビューで右クリックすることで表示されるコンテキスト メニューから行われます(*以下のスクリーンショットを参照*)。コンテンツ モデルビューで指定された場所で許されている処理のコマンドだけが有効になります。操作は右クリックされたオブジェクトに対して相対的に行われます。例えば「子の追加」コマンドを選択すると、右クリックされたオブジェクトに対する子が追加されます。

コンテキスト メニューから利用可能な操作のリストを以下に示します:

- *子コンポジッターとコンポーネントの追加:* **「子の追加」**コマンドは追加するコンポジッターまたはコンポーネントを選択することができるサブメニューを開きます。

- *コンポーネントとコンポジッターの挿入/追加:* 選択されたオブジェクトと同じ階層にコンポジッターまたはコンポーネントを挿入します。「**挿入**」コマンドにより選択されたオブジェクトの前に、「**追加**」コマンドにより最後の兄弟オブジェクトの後へ追加されます。

- *コンポジッターの変更:* コンポジッターを右クリックして、「**モデルの変更 | <新たな種類のコンポジッター>**」を選択してください。

- *グローバルコンポーネントの作成:* (i)「**新規グローバル**」コマンドはコンテンツモデルビューのあらゆる場所からアクセスすることができます。作成される新規のグローバルコンポーネントを選択することのできるサブメニューが表示されます。(ii) オブジェクトがグローバルコンポーネントとして作成可能な場合、コンテキスト メニューで「**グローバル化**」コマンドが有効になります。このコマンドを選択することで、オブジェクトがグローバルコンポーネントとして作成されます。コンテンツモデルビュー内で新規に作成されたグローバルコンポーネントへの参照が含まれます。

- *出現定義の変更:* 「**省略可能**」ならびに「**無制限**」コマンドを組み合わせることで希望する出現設定を得ることができます: (i) *「省略可能」* = 0 または1、(ii) *「省略可能」* + *「無制限」* = 0 から無限、(iii) *「無制限」* = 1 から無限、(iv) *なし* = 1 のように指定できます 。(メモ *「省略可能」* によりコンポーネントの `minOccurs` 属性が、*「無制限」* により `maxOccurs` 属性が影響を受けます)。

- *ローカルとグローバル定義の切り替え:* ローカル要素と同じ名前のグローバル要素が存在する場合、「**参照**」コマンドによりグローバル要素への参照と、ローカルの定義を切り替えることができます。

- *別のコンテンツモデルへジャンプ:* コンポーネントが自身のコンテンツモデルを持つ場合 (例えば、グローバルコンポーネントを参照する場合)「**定義へ移動**」コマンドが有効化され、コンテンツモデルへ移動することを選択できるようになります。代わりに「Ctrl」キーを押下しながらコンポーネントをダブルクリックしてもジャンプすることができます。

- *コンポーネントの型定義に移動:* コンポーネントが型定義 (単純型または複合型), を持つ場合「**型定義へ移動**」コマンドを使用すると型定義に移動することができます。ビルトインの単純型の場合、単純型に関する情報が含まれたメッセージボックスが表示されます。

- *定義済みの詳細情報を編集:* 定義済みの詳細が(スキーマ設計ツールバーの**定義済みの詳細を追加**アイコンにて) 図の中に表示されるようセットされている場合、「**編集**」コマンドは定義済みの詳細を含むサブメニューを表示します。定義済みの詳細を選択することで、その値を図の中で編集することができます。

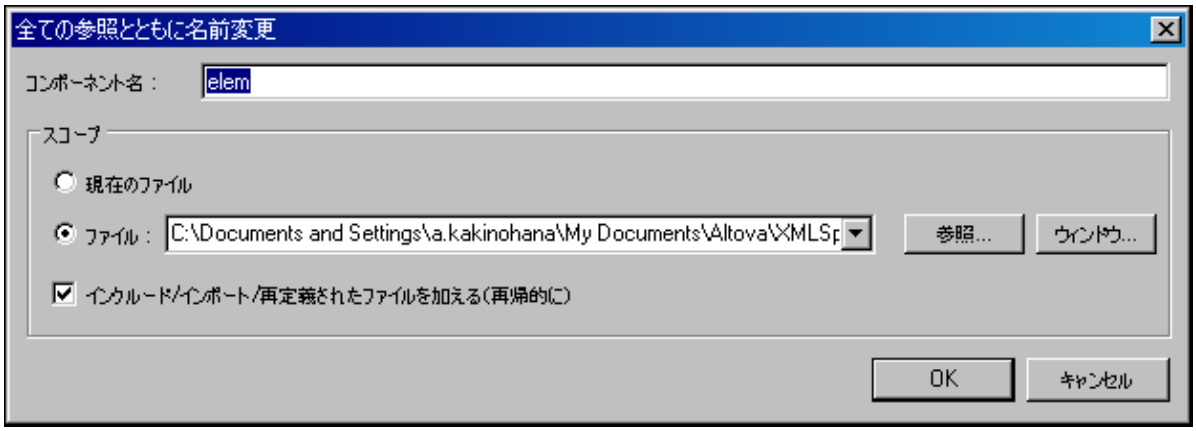
- *コンポーネントとコンポジッター注釈の作成:* 「**注釈の編集**」 コマンドにより、コンポジッター/コンポーネントの下部に注釈を作成するための空間が作成されます (*以下のスクリーンショットを参照*)。この空間では注釈の入力と編集を行うことができます。注釈が既に存在する場合、コメントをクリックすることで注釈テキストがハイライトされ編集することができるようになります。既存の注釈テキストをダブルクリックすることでも編集を行うことができます。

XML スキーマドキュメントでは、コンポジッターまたはコンポーネントの 注釈/ドキュメント 要素内部に注釈が作成されます。*以下にあるドキュメンテーションに関する記述も参照ください。*

- *インスタンスノードの XPath をコピー:* 「**インスタンスノードの XPath をコピー**」 コマンドは、グローバル要素ならびにグローバルな複合型にて定義された要素と属性に対して有効になります。 このコマンドにより、選択されたノードを指定するための XPath 条件式がクリップボードへコピーされます。path 条件式は、コンテンツモデルビューに現在表示されているグローバルコンポーネントから開始されます。

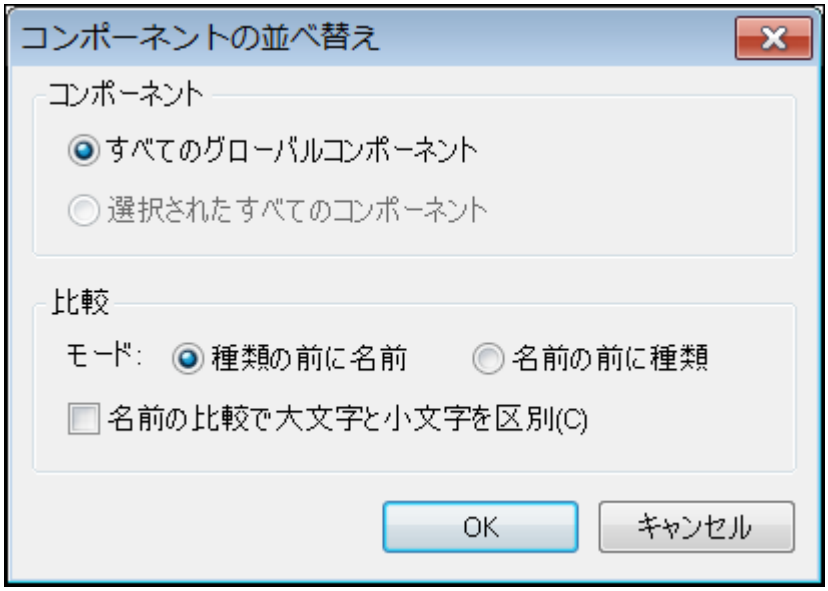
- *コンポーネントの検索と名前変更:* The commands「**すべての参照を検索**」ならびに「**すべての参照とともに名前変更**」コマンドはグローバル要素に対して有効になります。これらのコマンドは、アクティブなドキュメント、またはアクティブなドキュメントに関連するすべてのドキュメントにて選択されたコンポーネントの全出現を検索し、選択されたコンポーネントの全出現を名前変更します。

 (C) 2015-2021 Altova GmbH



上のスクリーンショットでは、右クリックされたコンポーネントと、選択されたスコープ内におけるそのコンポーネントへの参照名が `Email` に置き換えられます。詳細については「グローバルの検索と名前変更」317 を参照ください。

- *宣言と参照の並べ替え*: 「**並べ替え**」コマンドにより、選択されているすべてのコンポーネントまたは、選択されたコンポーネントの兄弟を並べ替えることができます。コンポーネントの並べ替えダイアログ（*下のスクリーンショット*）にて設定を行い、「OK」をクリックしてください。



「**Shift**」または「**Ctrl**」キーを押下しながらクリックすることで、複数のコンポーネントを選択することができます。コンポーネント名とその種類を並び替えのキーとして使用することができます。

メモ　コンポーネントをコピー、切り取り、削除、およびドラッグすることができます。一部の場合 `complexType` 制約のこのような属性は許可されない可能性があります。

キーボード ショートカットとドラッグ アンド ドロップ

キーボードの**Ctrl+c** ならびに**Ctrl+v** ショートカットを使用することで、コンテンツ モデルビュー内にある要素のコピーと貼り付けをそれぞれ行うことができます。構造的な理由からコピーや貼り付けが行えない場合、その旨のメッセージが表示されます。

その他にも、ドラッグ アンド ドロップにより（i）オブジェクトを図の別の場所へ移動、（ii）属性などのコンポーネントを入力ヘルパーから図の中へコピーすることができます。

コンポーネント プロパティ

コンポーネントのボックス内に定義済みの詳細が表示されるようにコンテンツ モデルビューが設定されている場合、図の中でこれら詳細情報を直接編集することができます。定義済みの詳細表示は、「**定義済みの詳細を追加**」ツール バー アイコンをクリックすることで切り替えることができます（*上記のアイコン リストを参照ください*）。

表示設定がなされている場合、コンポーネントのプロパティを詳細入力ヘルパー205で変更することで、その変更がプレースホルダ フィールドに反映されます。

注釈

XML スキーマの注釈は `annotation` 要素内部に含まれます。注釈には 2 つの種類があり、それぞれ `annotation` の子要素に含まれます:

- `documentation` 子要素: スキーマの編集者にとって有用な情報が含まれます。
- `appinfo` 子要素: 処理を行うアプリケーションが使用することのできるスクリプトや情報を挿入することができます。

両方の子要素を含んだ annotation 要素の例が以下に示されています。

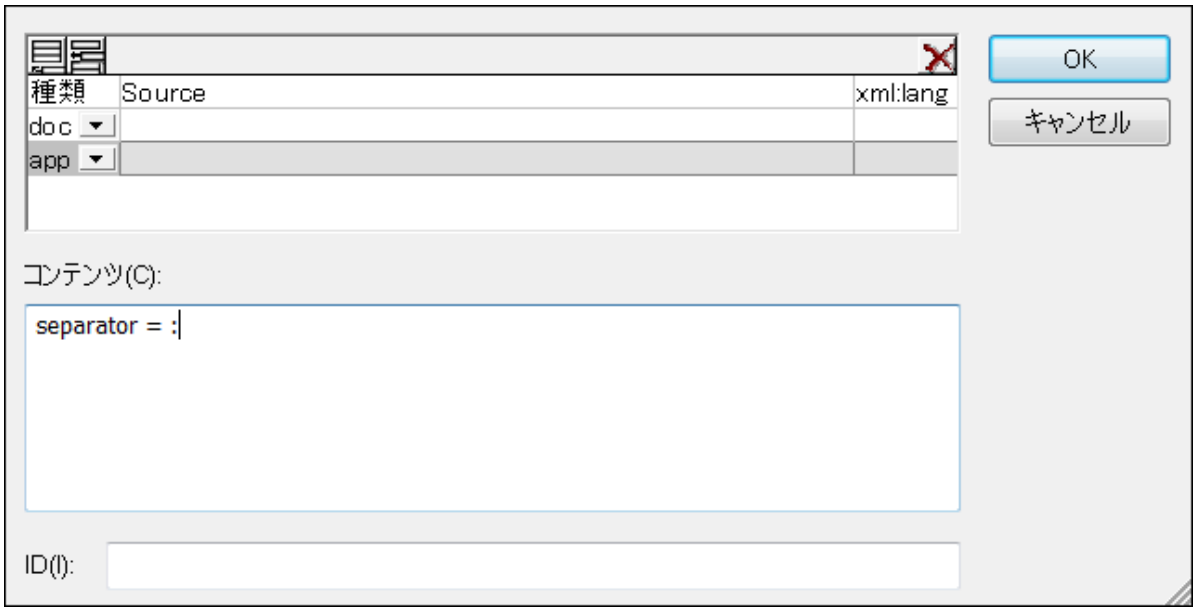
```
<xs:element name="session_date" type="xs:dateTime" nillable="true">
   <xs:annotation>
      <xs:documentation>Date and time when interview was held</xs:documentation>
      <xs:appinfo source="http://www.altova.com/datehandlers/interviews">separator =
:</xs:appinfo>
   </xs:annotation>
</xs:element>
```

コンテンツ モデルビューでは、個別のコンポジターとコンポーネントのために注釈を以下のように作成することができます。

1. コンポジターまたはコンポーネントを右クリックします。

2. コンテキスト メニューから、「**すべての注釈データ**」を選択します。注釈ダイアログ ボックスが表示されます（*以下のスクリーンショットを参照*）。その要素に対して注釈（`documentation` または `appinfo`）が既に存在する場合、ダイアログの対応する行に、それらの情報が表示されます。

(C) 2015-2021 Altova GmbH



3. appinfo 要素を作成するには、ダイアログの左上にある追加 または挿入 アイコンをクリックしてください。
4. 新たに作成された行の*種類*フィールドから app オプションを選択してください。
5. ダイアログのコンテンツ ペインに、アプリケーションにより処理されるスクリプトや情報を記述します。
6. オプションとして、処理アプリケーションが参照することのできるソース URI をソースフィールドに入力することもできます。

コメントと処理命令

XML スキーマドキュメントが XMLSpy 内部にロードされるまたは変更されると、単純型ならびに複合型内にあるコメントと処理命令が、それらを含んでいるオブジェクトの最後部にまとめて移動されます。このような移動を望まない場合、コメントの代わりに注釈を使用する必要があります。

コンテンツ モデルのドキュメント化

スキーマに関する詳細なドキュメンテーション[868]を HTML ならびに MS Word フォーマットで生成することができます。各グローバルコンポーネントに関する詳細なドキュメンテーションが生成され、目次ページに表示されたグローバルコンポーネントのリストから、個々のコンポーネントへ素早くアクセスすることができます。更に、関連する要素（子要素や複合型など）もハイパーリンクにより参照され、要素から要素へのナビゲーションを簡単に行うことができます。スキーマのドキュメンテーションを生成するには、メニューコマンドから「**スキーマ設計 | ドキュメンテーションの生成**」を選択してください。

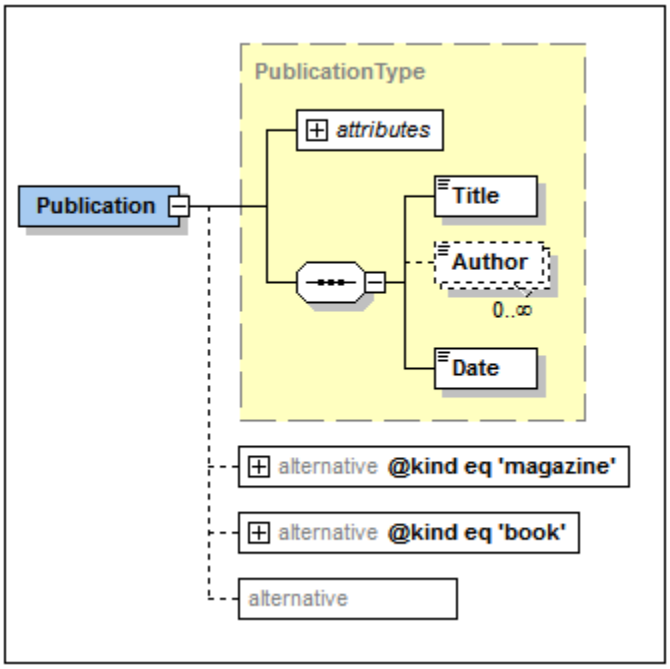
4.4.3.3 条件型割り当て

条件型割り当ては XSD 1.1 に新たに追加された機能で、要素に含まれる属性値や属性が存在するかにより XML ドキュメント内にある要素の型が決定されます。以下の様な XML ドキュメントを例に考えてみましょう:

```
<publication kind="magazine">
        ...
</publication>
```

スキーマでは、インスタンス要素の kind 属性値により、複数の publictaion 要素の型を指定することができます。スキーマでは、XSD 1.1 にて新たに追加された alternative 要素が使用されます。各 alternative 要素では、それぞれ異なる型が指定されます。

以下のスクリーンショットでは、test 属性を含むalternative 子要素（@kind eq 'magazine' と@kind eq 'book'）とともにPublication 要素が宣言されています。3 番目のalternative 要素にはtest 属性が含まれておらず、（詳細入力ヘルパーで割り当てられており、図の中には表示されていない）xs:error の単純型が割り当てられており、必要なときにXML 検証エラーが返されます。



上の宣言を以下に示します:

```
<xs:complexType name="PublicationType">
	<xs:sequence>
		<xs:element name="Title" type="xs:string"/>
		<xs:element name="Author" type="xs:string" minOccurs="0"
maxOccurs="unbounded"/>
		<xs:element name="Date" type="xs:gYear"/>
	</xs:sequence>
	<xs:attribute name="kind" type="xs:string"/>
</xs:complexType>

<xs:complexType name="MagazineType">
	<xs:complexContent>
		<xs:restriction base="PublicationType">
		<xs:sequence>
				<xs:element name="Title" type="xs:string"/>
				<xs:element name="Date" type="xs:gYear"/>
			</xs:sequence>
		</xs:restriction>
	</xs:complexContent>
</xs:complexType>

<xs:element name="Publication" type="PublicationType">
	<xs:alternative test="@kind eq 'magazine'" type="MagazineType"/>
	<xs:alternative test="@kind eq 'book'">
		<xs:complexType>
			<xs:complexContent>
				<xs:extension base="PublicationType">
					<xs:sequence>
```

```
                                        <xs:element name="ISBN" type="xs:string"/>
                                        <xs:element name="Publisher" type="xs:string"/>
                                    </xs:sequence>
                                </xs:extension>
                            </xs:complexContent>
                        </xs:complexType>
                    </xs:alternative>
                    <xs:alternative type="xs:error"/>
</xs:element>
```

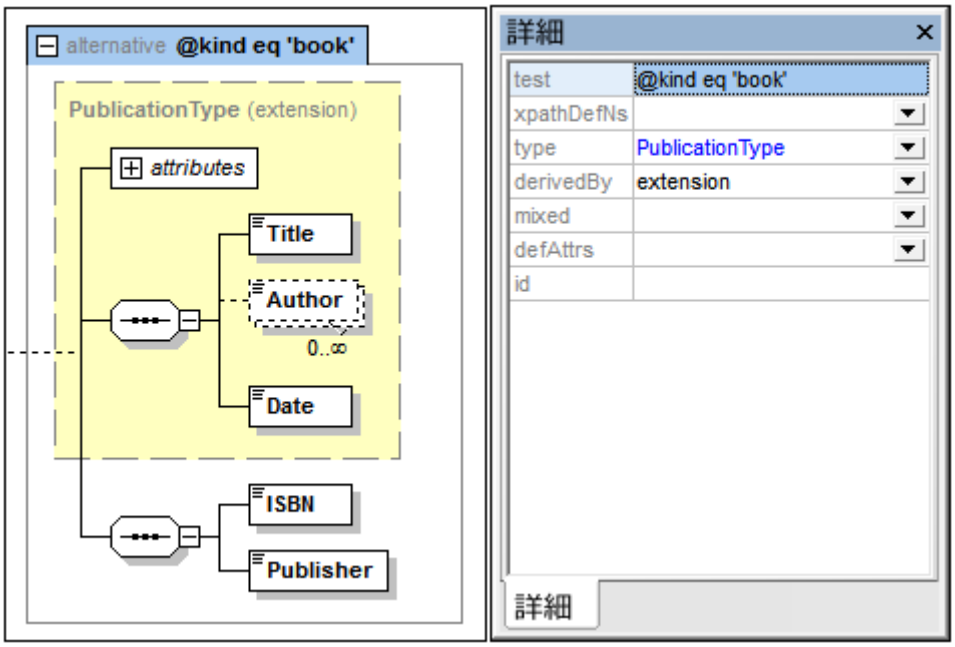
以下の点に注目してください：

- 複数ある alternative 要素の中で、最初に true と評価された alternative 要素が選択されるため、alternative 要素の順序は重要な意味を持ちます。コンテンツ モデル ビューでは、alternative 要素のボックスをドラッグすることで、その順序を変更することができます。
- Publication 要素には型（PublicationType）が含まれていることに注目してください。どの alternative 要素も使用されない場合、この型がデフォルトの型として使用されます。上の例では、最初の 2 つの alternative が false を返した場合、xs:error 型の alternative 要素が使用されます。
- どの alternative 要素条件も満たされず、デフォルトの型も定義されていない場合、要素は anyType 型に割り当てられます。これにより、XML コンテンツの整形式を保つことができる場合もあります。
- alternative 要素と単純型の xs:erroor は XSD 1.1 に新たに追加された要素です。

コンテンツ モデル ビューでの編集

要素のコンテキスト メニューから子として alternative 型を要素宣言に追加することもできます（以下のスクリーンショットにあるコンテンツ モデルを参照）。

alternative 要素の型は詳細入力ヘルパーで指定されます。複合型の場合、それが alternative 要素の展開された長方形にて表示され、編集することができます（以下のスクリーンショットを参照）。単純型は図に表示されませんが、詳細入力ヘルパーの単純型タブにて定義することができます。



メモ　詳細入力ヘルパーでは、XPath 条件式の名前空間を xpathDefaultNamespace 属性にて指定することができます。XPath デフォルト名前空間に関する詳細については、以下の記述を参照ください。

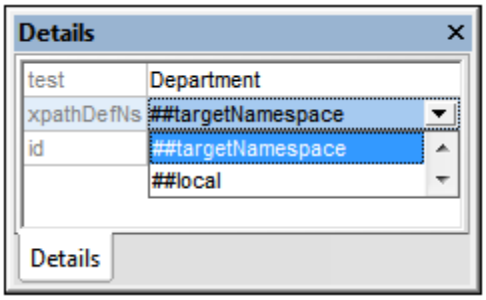
xpathDefaultNamespace の使用

XML スキーマドキュメントで宣言されたデフォルト名前空間は、そのXML スキーマドキュメントのデフォルト名前空間です。スキーマドキュメント内の接頭辞のない要素名に対して適用されますが、スキーマドキュメント内の接頭辞のないXPath 条件式に対しては適用されません。

XSD 1.1 にて新たに追加された xpathDefaultNamespace 属性（ XSD 1.1 の新規機能）は、XPath 条件式にて接頭辞のない要素の名前空間を指定するために使用されます。XPath デフォルト名前空間は、それらが宣言されたXML スキーマ要素上でスコープされます。xpathDefaultNamespace 属性は、以下のXML Schema 1.1 要素内に記述することができます:

- xs:schema
- xs:assert とxs:assertion
- xs:alternative
- xs:selector とxs:field（同一性制約にて）

XSD 1.1 モードでは、xs:schema 上のxpathDefaultNamespace がスキーマ設定ダイアログ（**「スキーマ設計 | スキーマ設定」**）にてセットされます。上に示されるその他の要素の場合、**xpathDefaultNamespace 属性が詳細入力ヘルパーにてセットされます（以下のスクリーンショットを参照）**。xs:schema 上でXPath デフォルト名前空間を宣言することで、スキーマ全体に対するXPath デフォルト名前空間が宣言されます。この宣言は、xpathDefaultNamespace 属性が許されている属性でオーバーライドすることができます（*上のリストを参照ください*）。



Xs:schema 上でXPath デフォルト名前空間を宣言することで、スキーマ全体に対するXPath デフォルト名前空間が宣言されます。この宣言は、xpathDefaultNamespace 属性が許されている属性でオーバーライドすることができます（*上のリストを参照ください*）。

実際の名前空間を含める代わりにxpathDefaultNamespace 属性には、以下のうちのどれかを含めることができます:

- ##targetNamespace: スキーマのターゲット名前空間と同じ名前空間がXPath デフォルト名前空間として使用されます。
- ##defaultNamespace: スキーマの名前空間と同じ名前空間がXPath デフォルト名前空間として使用されます。
- ##local: XPath デフォルト名前空間は指定されません。

ドキュメントに対してXPath デフォルト名前空間が宣言されなかった場合、XPath 条件式における接頭辞なしの要素は名前空間を持たない要素として扱われます。

メモ XPath デフォルト名前空間宣言は属性に対して適用されません。

(C) 2015-2021 Altova GmbH

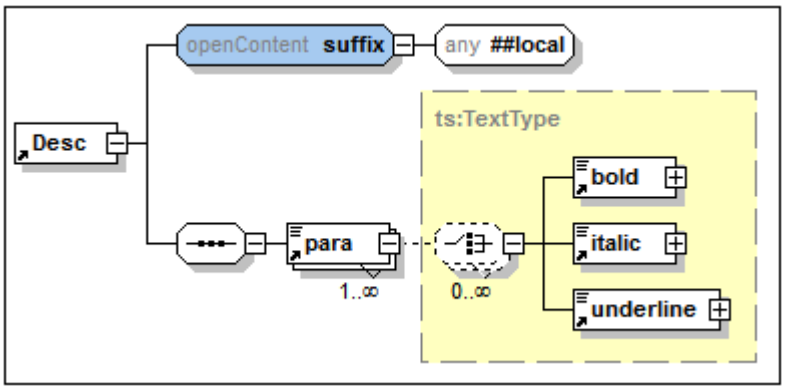
4.4.3.4 オープン コンテンツ モデル

オープン コンテンツはXSD 1.1 に新たに追加された機能で、複合型にて宣言され、(i) コンテンツ モデルにて定義された要素の間に、または(ii) コンテンツ モデルで定義された要素の後に記述されるあらゆる種類の要素（複合型のコンテンツ モデルでは定義されていない要素）を許可するために使用されます。

openContent 要素は複合型の子要素で、複合型のコンテンツ モデル前に一度出現します（以下のスクリーンショットを参照）。

モード

openContent 要素には必須のmode 属性が含まれており、値にはinterleave、suffix、またはnone のいずれかが使用されます（以下のスクリーンショットを参照）。デフォルトではinterleave が使用されます。



これら値が持つ意味を以下に記します:

- mode="interleave" またはmode="suffix" となっている場合、最低ならびに最大出現数が指定されていないワイルドカード要素コンテンツ(xs:any) が必要になります。これにより、任意の数の未定義要素（ワイルドカード）が許されることになります。
- モードがinterleave の場合、コンテンツ モデルにて定義された個々の要素の前または後に任意の数の未定義要素が出現できるようになります。
- モードがsuffix の場合、コンテンツ モデルにて定義されている最後の要素の後に任意の数の未定義要素が出現できるようになります。
- モードがnone の場合、未定義の要素（xs:any 子要素）の出現は許されず、コンテンツ モデルもオープンとはなりません。none 値は、スキーマ全体をスコープにするdefaultOpenContent 161 要素をオーバーライドするために使用されます。

コンテンツ モデル ビューでは、複合型の子要素として（コンテキスト メニューの「子の追加」から）オープン コンテンツが追加されます。図の中にあるopenContent ボックスをダブルクリックして値（interleave、suffix、またはnone）を選択するか、詳細入力ヘルパーからモードの値を選択してください。

ワイルドカード（xs:any）プロパティ

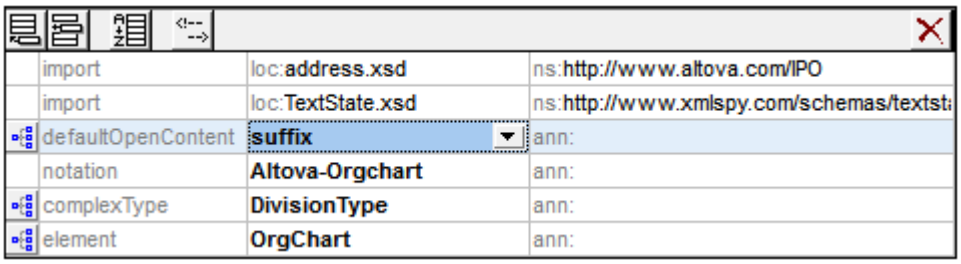
ワイルドカード プロパティは、ワイルドカードの詳細入力ヘルパーにて指定されます。図の中でワイルドカードを選択し、詳細入力ヘルパーにてプロパティ値を入力してください。

デフォルト オープン コンテンツ

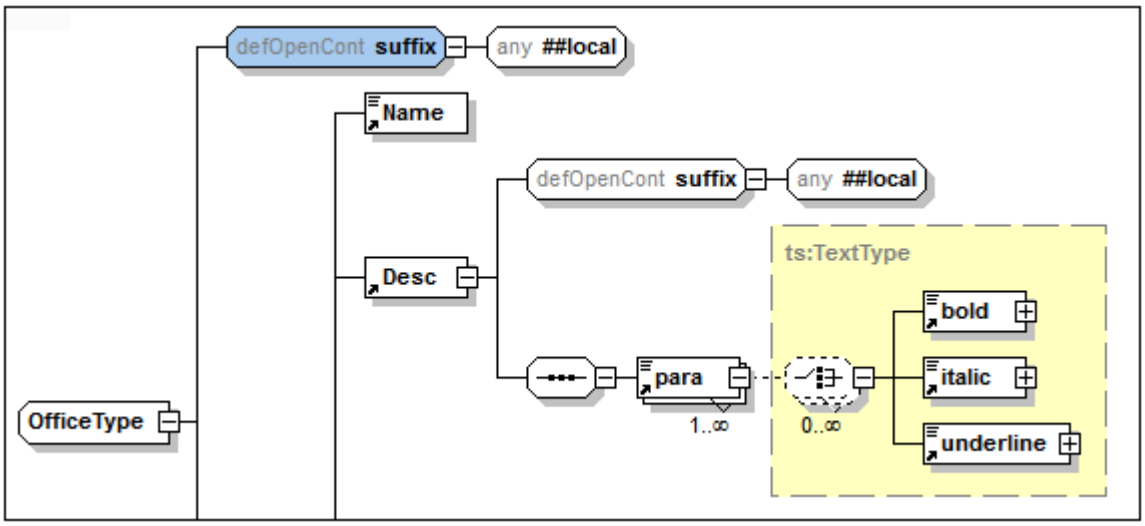
defaultOpenContent 要素はXSD 1.1 にて新たに追加された機能で、混合または子要素だけが許されている複合型に追加することのできる未定義の要素を指定するために使用されます。同じくXSD 1.1 で新たに追加されたopenContent 要素に似ていますが、

openContent 要素が単一の複合型に対して適用されるのに対して、defaultOpenContent 要素はスキーマ内にあるすべての複合型に対して適用されます。

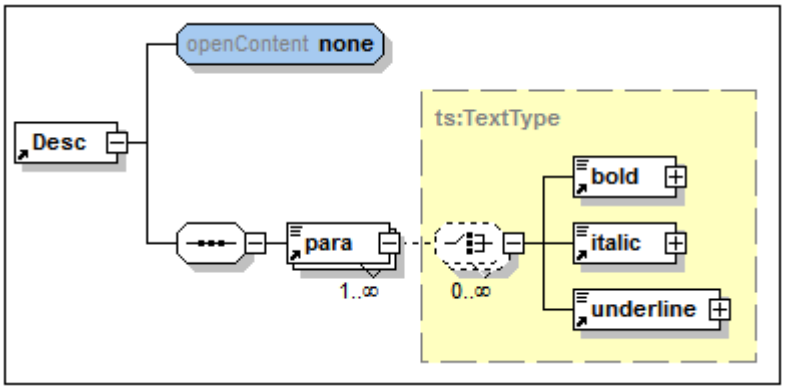
defaultOpenContent 要素はグローバル要素 161 で、ドキュメント内（インクルード、インポート、再定義の後、そしてコンテンツ定義の前）に一度だけ出現することができます（以下のスクリーンショットを参照）。要素には interleave または suffix という値を含めることのできる mode 属性が含まれており、デフォルトでは interleave 値が使用されます。



defaultOpenContent 要素には openContent 要素の定義と同様に、コンテンツ モデルビューで編集することのできるコンテンツ モデルが含まれています（上記参照）。一旦 defaultOpenContent 要素が宣言されると、スキーマ内にあるすべての複合型に対して自動的に適用され、各複合型のコンテンツ モデルに表示されます。以下のスクリーンショットでは、defaultOpenContent が OfficeType と Desc という複合型に対して自動的に適用されているのが確認できます。

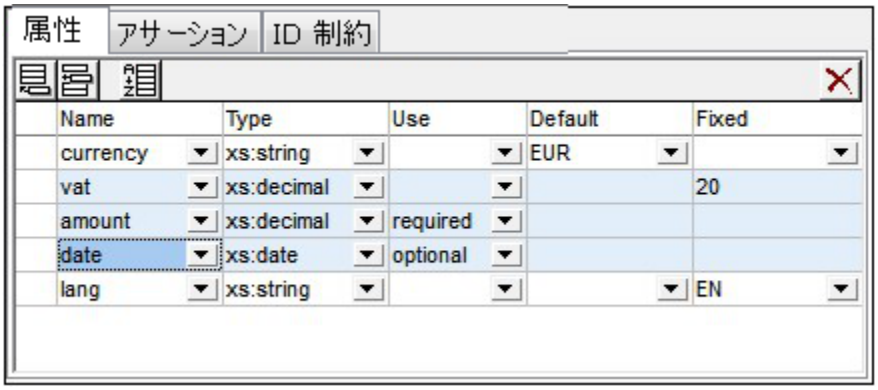


特定の複合型に対して defaultOpenContent 要素をオーバーライドするには、openContent 子要素をその複合型へ追加してください。以下のスクリーンショットでは、defaultOpenContent が適用された Desc 要素以下に openContent 子要素が定義されており、defaultOpenContent 要素がオーバーライドされています。



4.4.4　属性、アサーション、同一性制約

属性/アサーション/同一性制約（AAIDC）ペインは、スキーマ概要メイン ペインならびにコンテンツ モデルビューの下部に表示されます（以下のスクリーンショットを参照）。ペインとそのタブは固定されていますが、コンテンツ モデルビューでは各タブの表示内容を（AAIDC タブではなく）コンテンツ モデルビューに表示して、図の中で直接編集することができます。3 つの タブの内容を図の中に表示するように指定すると、AAIDC ペインが表示されなくなります。



AAIDC ペインと図の中における表示は、スキーマ表示設定ダイアログ[873]（「**スキーマ設計 | 表示設定 | 要素タブ**」）または（以下に示される）スキーマ設計ツール バーのアイコンから切り替えることができます。

	属性を図に表示: コンテンツ モデルビューが表示されている際にスキーマ設計ツール バーで有効になります。属性の表示を図または属性タブで行うか指定します。
	アサーションを図に表示: コンテンツ モデルビューが表示されている際にスキーマ設計ツール バーで有効になります。アサーションの表示を図またはアサーション タブで行うか指定します。.
	制約を図に表示: コンテンツ モデルビューが表示されている際にスキーマ設計ツール バーで有効になります。同一性制約の表示を図または同一性制約タブで行うか指定します。.

タブの使用

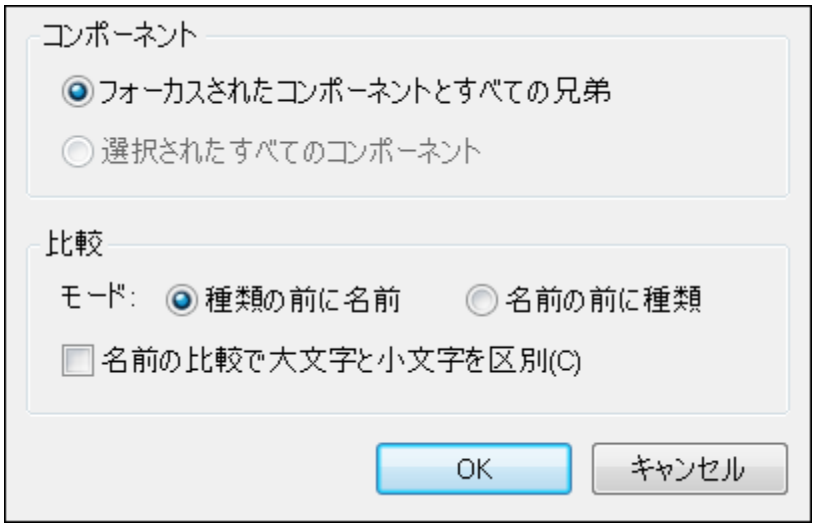
AAIDC ペイン内部のタブは、スキーマ概要またはコンテンツ モデルビューにて選択されているコンポーネントの種類に対応しているタブが有効になります。例えば複合型に対して属性を追加することはできるため、メイン ペインにて複合型が選択されている時に属性タブが有効になります（タブが有効になるとは、そのタブの中でなんらかの操作ができることを意味します）。

各タブの使用方法については、このセクション以下にあるサブセクションを参照ください：

- 属性、属性グループ、属性ワイルドカード 188
- アサーション 191
- 同一性制約 194

属性と同一性制約の並べ替え

各タブのツール バーにある並べ替えアイコンをクリックすることで、属性と同一性制約の並べ替えを行うことができます。コンポーネントの並べ替えダイアログでは、選択されたコンポーネントとその兄弟、または選択されたコンポーネントだけを並べ替えの対象とするかを選択することができます。例えば上のスクリーンショットでは、3 つの属性が選択（ハイライト）されています。「Shift」キーを押下しながらクリックすることで範囲の選択を、「Ctrl」キーを押下しながらクリックすることで、コンポーネントを選択対象に加えることができます。



コンポーネントのセットを選択した後には最初にアルファベット順で並べ替えを行い、その次に種類による並べ替えを行うのか（種類の前に名前）、またはその逆（名前の前に種類）といった、並べ替えの優先順位を選択することができます。並べ替えの結果はスキーマドキュメントのテキストへ直ちに反映されます。

4.4.4.1 属性、属性グループ、属性ワイルドカード

属性/アサーション/同一性制約（AAIDC）ペインの属性タブでは、以下の操作を行うことができます:

- 選択された複合型にて、ローカルの属性を宣言 189
- 選択された複合型で使用する属性グループの参照 190
- 選択された複合型にて属性ワイルドカードを定義 190

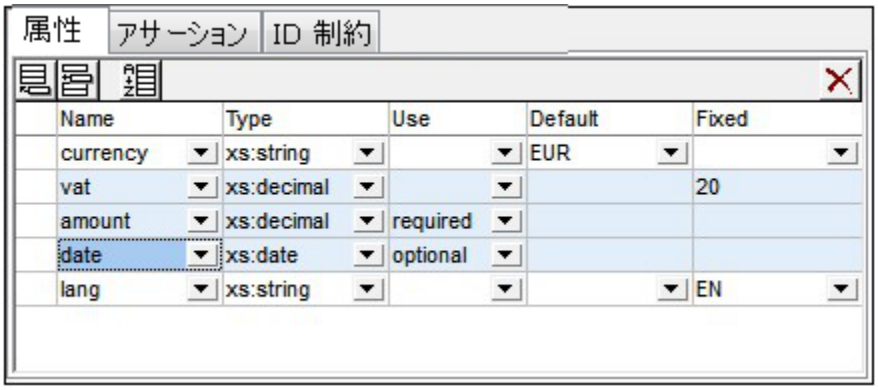
(C) 2015-2021 Altova GmbH

メモ　AAIDC ペインではなく、図の中に属性を表示するオプションを選択した場合（「**スキーマ設計 | 表示設定**」）、属性、属性グループ参照、属性ワイルドカードのプロパティを、図の中、そして詳細入力ヘルパーにて編集することができます。

	属性を図に表示: コンテンツモデルビューが表示されている際にスキーマ設計ツールバーで有効になります。属性の表示を図または属性タブで行うか指定します。

属性

属性/アサーション/同一性制約（AAIDC）ペインの属性タブでは、要素や複合型のローカル属性、または属性グループを構成する属性を宣言することができます（以下のスクリーンショットを参照）。



以下の操作により属性を作成することができます:

1. スキーマ概要にて複合型または属性を作成するための属性グループを選択します。
2. 属性タブの左上にある「**追加**」または「**挿入**」アイコンをクリックして、「**属性**」を選択します。
3. 属性に対して作成された行にて属性の詳細（Name、Type、Use、Default、Fixed の値）を入力します。Name プロパティは必須の項目で、`use` のデフォルト値はオプションです。データ型（Type）とDefault/Fixed 値はオプションです。

メモ　属性グループへの属性追加はスキーマ概要[151]でのみ行うことができますが、複合型への属性追加はスキーマ概要[151]とコンテンツモデルビュー[151]の両方で行うことができます。

メモ　属性が型定義を持つ場合、コンテキストメニューコマンド「**型の定義**」をクリックすると型定義に移動することができます。この場合、ビルトインの単純型の場合、単純型の情報を含むメッセージボックスが表示されます。

デフォルト値と固定値

属性宣言で宣言されたデフォルト値（Default）または固定値（Fixed）は、インスタンスドキュメント内にその属性が存在しない場合に（インスタンスドキュメントに対して）適用されます。指定できるのはデフォルト値と固定値の一方で、両方を指定することはできません（上のスクリーンショットを参照）。デフォルト値または固定値は属性のデータ型に対して妥当な値である必要があります。`use` に`required` がセットされている場合、デフォルト値も固定値も定義することはできません。

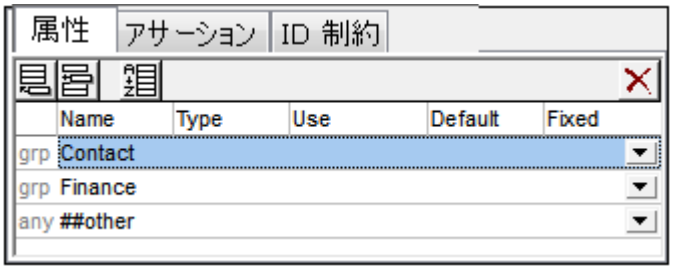
以下の点に留意してください:

- *デフォルトの値*: 属性が存在しない時だけデフォルト値は挿入されます。属性が存在しており、値が与えられている場合、デフォルト値は挿入されません。インスタンスドキュメント内の値が妥当ではない場合、エラーが返されます。
- *固定値*: 固定値は属性が存在しない場合だけではなく、インスタンスドキュメントの値が属性の宣言にて指定された固定値と異なる場合にも挿入されます。

メモ デフォルト値と固定値はローカルならびにグローバル属性に対しても指定することができます。ローカル属性の場合はAAIDC ペインの属性タブと詳細入力ヘルパーの両方で定義することができ、グローバル属性の場合、詳細入力ヘルパーにて定義することができます。

属性グループと参照

グローバルの属性グループが宣言されると、この属性グループに対する参照を複合型の定義内で追加することができます。スキーマ概要またはコンテンツ モデルビューにて複合型を選択し、AAIDC ペインの属性タブ左上にある「**追加**」または「**挿入**」アイコンをクリックして、「**属性グループ**」を選択してください。作成された属性グループ行にて参照される属性グループの名前を入力してください（以下のスクリーンショットには2 つの属性グループ参照が含まれています）。複数の属性グループを追加することができます。

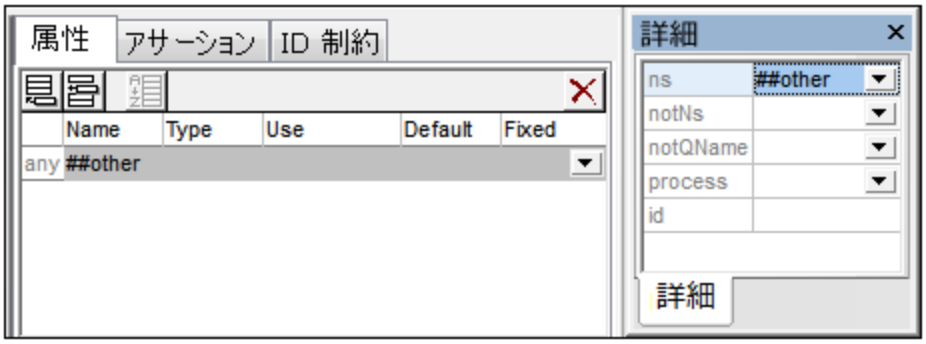


属性タブにて属性グループが選択されると、そのプロパティを詳細入力ヘルパーにて編集することができるようになります。

属性ワイルドカード: anyAttribute

属性ワイルドカードは、任意の属性を要素上で使用するのを許可するため複合型へ追加されます。属性ワイルドカードは単一の `anyAttribute` 要素により定義され、インスタンスドキュメント内にある要素にて指定された名前空間から得られる任意の数の属性が出現するのを許可します。

スキーマ概要またはコンテンツ モデルビューにて複合型コンポーネントを選択し、AAIDC ペインの属性タブ左上にある追加または挿入アイコンをクリックし、「**任意属性**」を選択することで属性ワイルドカードは作成されます。属性ワイルドカード `anyAttribute` に対する行が作成されます（以下のスクリーンショットを参照）。

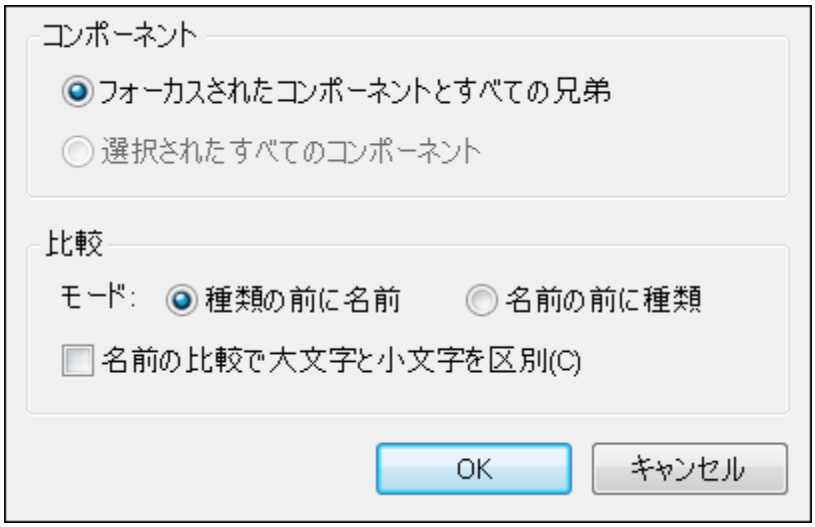


属性タブでは、`anyAttribute` の `namespace` プロパティをセットすることができます。その他のプロパティは、属性タブにて属性ワイルドカードが選択されている状態で詳細入力ヘルパーにてセットすることができます（上のスクリーンショットを参照）。`notNamespace` ならびに `notQName` プロパティはXSD 1.1 の機能 153で、XSD 1.0 モード 153では利用できない点に注意してください。

属性と属性グループの並べ替え

タブのツールバーにある「**並べ替え**」アイコンをクリックすることで、属性タブに表示されている属性や属性グループを並べ替えることができます。コンポーネントの並べ替えダイアログでは、単一のコンポーネントとその兄弟コンポーネント、または選択されているコンポーネントのセットを並べ替えの対象とすることができます（以下のスクリーンショットを参照）。「**Shift**」または「**Ctrl**」キーを押下しながらコンポーネントをクリックすることで、複数のコンポーネントを選択することができます。

 (C) 2015-2021 Altova GmbH



並べ替えを行う範囲を選択すると、属性すべてと属性グループをアルファベット順に並べ替え（種類の前に名前）、または属性と属性グループとアルファベット順で別々に並べ替えることができます（名前の前に種類）。

並べ替えの順序はスキーマドキュメントのテキストに直ちに反映されます。

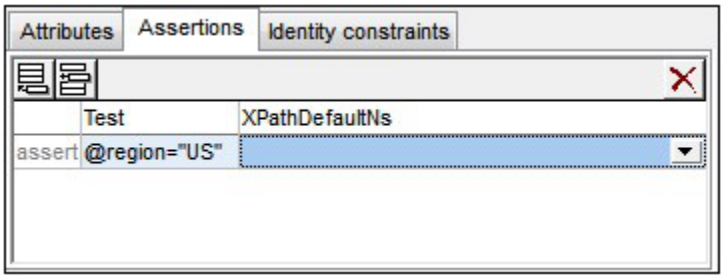
メモ　属性ワイルドカードは常に複合型宣言の末尾に出現する必要があり、複合型の定義に対して属性ワイルドカードはそれぞれ1つしか許されていないため、並べ替えの対象に含まれません。

4.4.4.2 アサーション

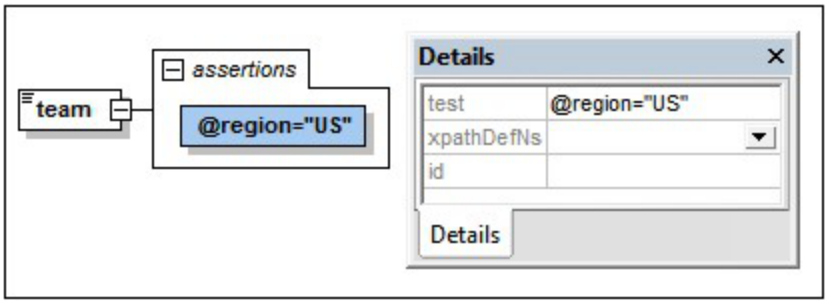
このセクションでは**複合型で使われるアサーション**について説明します。このようなアサーションは `xs:assert` 要素にて定義され、複合型の妥当性制約に使用されます（アサーションには `xs:assertion` 要素にて定義され、単純型のファセット入力ヘルパー207にて作成ならびに編集されるアサーションも存在します）。

スキーマビューでは、以下の場所で複合型のアサーションを作成し、編集することができます:

- *スキーマ概要にて:* 属性/アサーション/同一性制約（AAIDC）ペインのアサーション タブにて（以下のスクリーンショットを参照）。



- *コンテンツモデルビューにて:* アサーションは図の中に表示されたアサーション タブにて編集することができます（以下のスクリーンショットを参照）。図の表示オプションを選択するには、スキーマ デザイン ツールバーにある「**アサーションを図に表示アイコン**をクリックしてください。複合型や複合型の要素にあるアサーションのボックスを選択し、詳細入力ヘルパーにてアサーションの定義を編集してください。



	図内でアサーションを表示する: コンテンツモデルビューで有効化されます。図とアサーションタブ間のアサーションの表示を切り替えます。

メモ アサーションはXSD 1.1 に搭載されている機能で、アサーションの編集機能はXSD 1.1 モード 153でのみ有効になります。

アサーションのスコープ

アサーションの制約を定義するために使用されるXPath 条件式は、定義されている複合型のスコープ内にある必要があります。XPath 条件式が特定のノードへアクセスする必要がある場合、そのノードの祖先ノードにてアサーションを定義する必要があります。

アサーションの追加と削除

複合型には複数のアサーションを含めることができます。インスタンスドキュメント内にある要素に対してすべてのアサーションのXPath 条件式が`true` と評価されることで、ドキュメントも妥当なものと評価されます。以下の操作により、複合型に対してアサーションを追加することができます:

- **スキーマ概要にて**: 複合型を選択し、AAIDC ペインにあるアサーション タブの左上にある**追加**または**挿入**アイコンをクリックします（上のスクリーンショットを参照）。追加することのできるアサーションの数に制限はありません。アサーションを選択し、タブの右上にある**削除**アイコンをクリックすることで、そのアサーションを削除することができます。
- **コンテンツ モデル ビューにて**（上のスクリーンショットを参照): 複合型を右クリックして、「**子の追加 | アサーション**」を選択します。図に表示されている既存のアサーションを右クリックして、コンテキスト メニューから「**追加 | アサーション**」または「**挿入 | アサーション**」を選択することもできます。追加することのできるアサーションの数に制限はありません。アサーションを選択して、「**Delete**」キーを押下することで、そのアサーションを削除することができます。

アサーションのXPath 条件式を定義する

複合型アサーションのXPath 条件式により、インスタンスドキュメント内にある複合型要素に対して適用される妥当性制約が定義されます。例えば上のスクリーンショットでは、複合型要素の`team` とアサーションのXPath 条件式 `@region="US"` が定義されています。XML スキーマドキュメントで、アサーションは以下のように記述されます:

```
<xs:assert test='@region="US"'/>
```

アサーションにより、インスタンスドキュメント内にある`team` 要素には、`US` という値を持つた`region` 属性を含めなければならなくなります。この条件が満たされない場合、ドキュメントの検証に失敗します。

以下の点に留意してください:

- XPath 条件式はXPath 2.0 に従ったかたちで記述される必要があります。
- XPath 条件式で評価されるノードは、アサーションのスコープ内にある必要があります（上記を参照ください）。
- 条件式が`boolean` の`true/false` を評価しない場合、返された値が`boolean` 値に変換されます。空でないシーケンスは`true` に変換され、空のシーケンスは`false` に変換されます。

(C) 2015-2021 Altova GmbH

- 条件式における構文エラーは赤色で示されます。コンテキスト エラーは表示されません。例えばXPath 条件式にてある属性が評価され、その属性がスキーマ内部で定義されていなかったとしても、そのエラーは表示されません。

アサーションメッセージ

アサーションの詳細がこの定義と共に提供される場合とても役に立ちます。XML インスタンスドキュメントが検証される際、アサーションが満たされていない場合、適切な メッセージが表示されます。XML スキーマ 仕様は、このようなメッセージのために対応をしないため、XMLSpy は、アサーションの定義と共にXML インスタンスドキュメントの検証内で使用される `http://www.altova.com/ja/xml-schema-extensions` (または他の名前空間) 内でメッセージ を許可します。例えば

```
<xs:assert test="count(//MyNode) ge 1" altova:message="There must be at least one
MyNode element"/> or
<xs:assertion test="count(//MyNode) ge 1" altova:message="There must be at least one
MyNode element"/>
```

アサーション内で指定された制約が満たされない場合、XMLSpy の検証エンジンは、検証-エラーメッセージと共にヒントとしてのアサーションと関連したメッセージを表示します。バリデーターは、`assert/@message` 属性の値または、`message` 属性が存在する名前空間に関わらず、`assert/@message` 属性 を報告します。ですが、スキーマビュー内では、xml-schema-extensions 名前空間にある `message` 属性 のみを編集することができます。他の名前空間内の`message` 属性 を編集するには、テキストビューを使用します。

詳細に関しては、アサーションメッセージ 212 を参照してください。

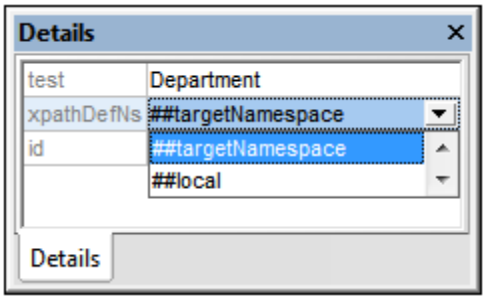
xpathDefaultNamespace の使用

XML スキーマドキュメントで宣言されたデフォルト名前空間は、そのXML スキーマドキュメントのデフォルト名前空間です。スキーマドキュメント内の接頭辞のない要素名に対して適用されますが、スキーマドキュメント内の接頭辞のないXPath 条件式に対しては適用されません。

XSD 1.1 にて新たに追加された `xpathDefaultNamespace` 属性 (XSD 1.1 の新規機能) は、XPath 条件式にて接頭辞のない要素の名前空間を指定するために使用されます。XPath デフォルト名前空間は、それらが宣言されたXML スキーマ要素上でスコープされます。`xpathDefaultNamespace` 属性は、以下のXML Schema 1.1 要素内に記述することができます:

- `xs:schema`
- `xs:assert` と`xs:assertion`
- `xs:alternative`
- `xs:selector` と`xs:field` (同一性制約にて)

XSD 1.1 モードでは、`xs:schema` 上の`xpathDefaultNamespace` がスキーマ設定ダイアログ (「スキーマ設計 | スキーマ設定」) にてセットされます。上に示されるその他の要素の場合、xpathDefaultNamespace 属性が詳細入力ヘルパーにてセットされます (以下のスクリーンショットを参照)。xs:schema 上でXPath デフォルト名前空間を宣言することで、スキーマ全体に対するXPath デフォルト名前空間が宣言されます。この宣言は、`xpathDefaultNamespace` 属性が許されている属性でオーバーライドすることができます (上のリストを参照ください)。



Xs:schema 上でXPath デフォルト名前空間を宣言することで、スキーマ全体に対するXPath デフォルト名前空間が宣言されます。この宣言は、xpathDefaultNamespace 属性が許されている属性でオーバーライドすることができます（*上のリストを参照ください*）。

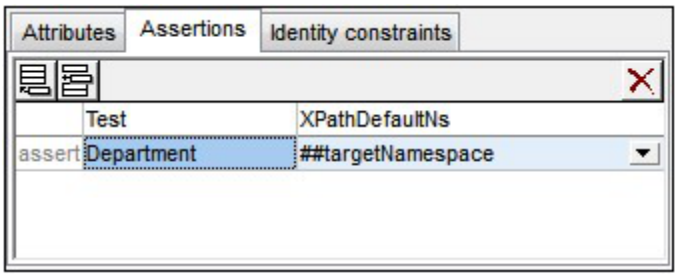
実際の名前空間を含める代わりにxpathDefaultNamespace 属性には、以下のうちのどれかを含めることができます:

- ##targetNamespace: スキーマのターゲット名前空間と同じ名前空間がXPath デフォルト名前空間として使用されます。
- ##defaultNamespace: スキーマの名前空間と同じ名前空間がXPath デフォルト名前空間として使用されます。
- ##local: XPath デフォルト名前空間は指定されません。

ドキュメントに対してXPath デフォルト名前空間が宣言されなかった場合、XPath 条件式における接頭辞なしの要素は名前空間を持たない要素として扱われます。

メモ XPath デフォルト名前空間宣言は属性に対して適用されません。

アサーション内にあるXPath 条件式アサーションの定義にてXPath デフォルト名前空間を定義することもできます。属性/アサーション/同一性制約（AAIDC）ペインのアサーション タブにて、XPathDefaultNs フィールドのドロップダウン リストから目的のキーワードを選択してください。



選択された名前空間が、そのアサーションのスコープとなります。

4.4.4.3 同一性制約

同一性制約（IDC）はグローバル要素宣言にて定義されます。同一性制約によりノードの一意性が指定され、ユニークなノード間の正確な参照を行うことができます。

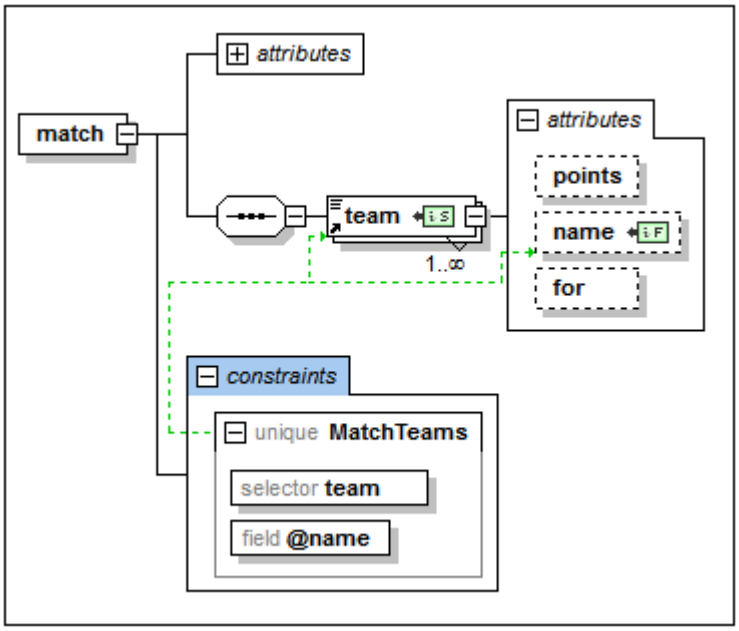
宣言のメカニズム

同一性制約（unique、key、keyref）を定義するために以下のメカニズムを使用することができます:

- スキーマ概要 156 に表示されたグローバル要素にて同一性制約を宣言することができます。グローバル要素を選択し、属性/アサーション/同一性制約（AAIDC）ペインにある同一性制約タブをクリックします（以下のスクリーンショットを参照）。

同一性制約タブにある挿入または追加アイコンにより、同一性制約（unique、key、keyref）を追加します。選択された同一性制約に対して field を追加することもできます。選択された field や同一性制約を削除するには、削除アイコンを使用してください。

- グローバル要素のコンテンツモデルビュー[168] では、グローバル要素やローカルの子孫要素にて同一性制約を定義することができます。このビューでは、同一性制約のタブと図の中に表示された要素の制約ボックスで同一性制約を編集することができます（以下のスクリーンショットでは、match 要素に team セレクターが含まれるというユニークな制約が示されています）。「**スキーマ設計 | 表示設定**」により表示されるスキーマ表示設定にて、またはスキーマ設計ツールバーにある「**制約を図に表示**」アイコンをクリックすることで、表示方法を切り替えることができます。図に示された同一性制約に対してドラッグ アンド ドロップ編集機能を使用することもできます。



同一性制約（unique、key、keyref）を図の中に追加するには、制約を行う要素を右クリックして、コンテキスト メニューから「**子の追加 | <同一性制約の種類>**」を選択してください。同一性制約が図の中で選択されていると、フィールド アイテムが選択可能になります。「**Delete**」キーを押下することで、選択された field または同一性制約を削除することができます。

以下にあるいずれかの方法で、selector ならびに field ボックスに XPath 条件式を入力することができます: (i) キーボードから入力する、(ii) selector または field ボックスをクリックすると表示されるドロップダウン リストから目的のノードを選択する、または (iii) ターゲット ノードを selector または field ボックスへドラッグして、ボックスがハイライトされたところでドロップする（XPath 条件式が自動的に作成されます）。

メモ　コンポーネント入力ヘルパーにある同一性制約のタブではスキーマ内にあるすべての同一性制約の概要[201] を確認することができます。詳細については、以下のコンポーネント入力ヘルパーを参照ください。

同一性制約アイコン

	制約を図に表示: コンテンツ モデルビューが表示されている際にスキーマ設計ツールバーで有効になります。同一性制約の表示を図または同一性制約タブで行うか指定します。
	同一性制約の視覚化: コンテンツ モデルビューにて選択可能になります。同一性制約の情報の表示/非表示を切り替えます。

	セレクター ノード、フィールド ノード: ノードの図にて表示され、これら 2 つのアイコンにより、selector ならびに field の XPath 条件式により選択されたノードがそれぞれ表示されます。

同一性制約の可視化

同一性制約の可視化アイコン が有効になっている場合、同一性制約の情報が図の中で可視化されます。可視化が有効になっている場合、selector と field の XPath 条件式により選択されたノードがボックス内のアイコンにより示され、緑色の線によりセレクターとフィールドのボックスが接続されます（上のスクリーンショットを参照ください）。

同一性制約の可視化アイコンにより、スキーマビューにおける同一性制約の検証機能も切り替えられます。XPath 条件式または同一性制約が間違っている場合、エラーが赤色で、警告がオレンジ色で示されます。XML スキーマドキュメントの検証を行うと、エラーと警告がメッセージ ウィンドウに表示されます。

同一性制約の可視化アイコン を無効にすることで、検証を無効にすることができます。

XML の内容

このセクションの下部に示される同一性制約の例は、以下の妥当なインスタンスドキュメントをベースにしています:

```
<results xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:noNamespaceSchemaLocation="Scores.xsd">
   <!----- Groups ----->
   <group id="A">
         <team name="Brazil"/>
         <team name="Germany"/>
         <team name="Italy"/>
         <team name="Holland"/>
   </group>
   <group id="B">
         <team name="Argentina"/>
         <team name="France"/>
         <team name="England"/>
         <team name="Spain"/>
   </group>
   <!----- matches ----->
   <match group="A" date="2012-06-12" location="Munich">
         <team name="Brazil" for="2" points="3"/>
         <team name="Germany" for="1" points="0"/>
   </match>
   <match group="A" date="2012-06-12" location="Frankfurt">
         <team name="Italy" for="2" points="1"/>
         <team name="Holland" for="2" points="1"/>
   </match>
   <match group="B" date="2012-06-13" location="Munich">
         <team name="Argentina" for="2" points="3"/>
         <team name="France" for="0" points="0"/>
   </match>
   <match group="B" date="2012-06-13" location="Berlin">
         <team name="England" for="0" points="1"/>
         <team name="Spain" for="0" points="1"/>
   </match>
</results>
```


(C) 2015-2021 Altova GmbH

ユニーク制約（unique）

ユニーク制約により、要素または属性（要素や属性のセット）の値が定義されたスコープ内でユニーク（一意）でなければならないと指定されます。なんらかのリーグ戦を表している上のXML テキストでは、同じ名前のチーム同士が試合を行わないような制約が必要となるため、各 `match` 要素のスコープにて `team/@name` ノードの値がユニークであるような制約を作成します。以下の操作により制約を作成します:

1. スキーマ概要にて `match` 要素を選択します。`match` 要素が同一性制約のスコープとなります。
2. 同一性制約タブの左上にある「**追加**」または「**挿入**」アイコンをクリックして、メニューから「**ユニーク**」をクリックします。ユニーク制約の行が追加されます（以下のスクリーンショットを参照）。

3. 同一性制約の名前を入力します（上のスクリーンショットでは、`MatchTeams` という名前が与えられています）。
4. `team` 要素を選択するXPath 条件式を `Selector` 列に入力します。`match` 要素がコンテキスト ノードだということご注意してください。これで `team` 要素が同一性制約のセレクターとなり、ユニーク制約が適用されるノードとなります。
5. ユニークでなければならない `@name` ノードをField 列に入力します。ユニーク制約が入力されたノードの値に対して適用されます。

上記のユニーク制約により、各 `match` 要素のスコープ内にてすべての `team` 要素にユニークな `@name` 属性値が必要になります。

更にフィールドを追加して、ユニーク制約をチェックすることもできます。例えば `result` 要素にて定義されたユニーク制約により、ユニークな日付と場所の組み合わせがすべての `match` に含まれていることをチェックすることができます（同じ日付に同じ場所で複数の試合を行うことができなくなります）。ユニーク制約により、各 `match` 要素（セレクター）に対して `@date` と `@location` の値を組み合わせた値が `result` 要素のスコープ内部でユニークとなる必要があります。

上記の方法により `result` 要素に対してユニーク制約を定義します。`match` がセレクターとなり、`@date` と `@location` がフィールドとなります（以下のスクリーンショットを参照）。追加アイコンをクリックして「**フィールド**」を選択することで、2 番目のフィールドが追加されます。

メモ　同一性制約タブのRefer 列はキー参照（`keyref`）制約でのみ有効になります。

キー制約（key）

キー制約により、以下の制約が指定されます: (i) 要素や属性（または属性やスキーマのセット）の値が定義されたスコープ内でユニークであり、(ii) これらフィールド要素や属性がインスタンスXML ドキュメントに含まれている（従って省略可能な要素や属性をキー制約のフィールド

として選択すべきではありません)。キー制約により、ユニーク制約により与えられる制約に加え、指定されたフィールド要素/属性をXMLドキュメント内に含める必要があります。

既に登場した最初のユニーク制約と同様に、match 要素にて定義されたキー制約を以下のスクリーンショットに示します:

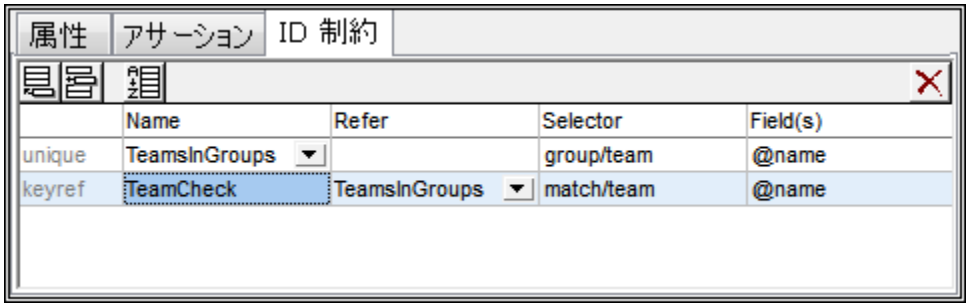
このキー制約により、各 match 要素のスコープ内部ですべての team 要素にユニークな @name 値が必要になります。更に、@name 属性がmatch/team 要素上に存在しなければならないとも指定されます。

メモ 同一性制約タブのRefer 列はキー参照(keyref) 制約でのみ有効になります。

キー参照(keyref)

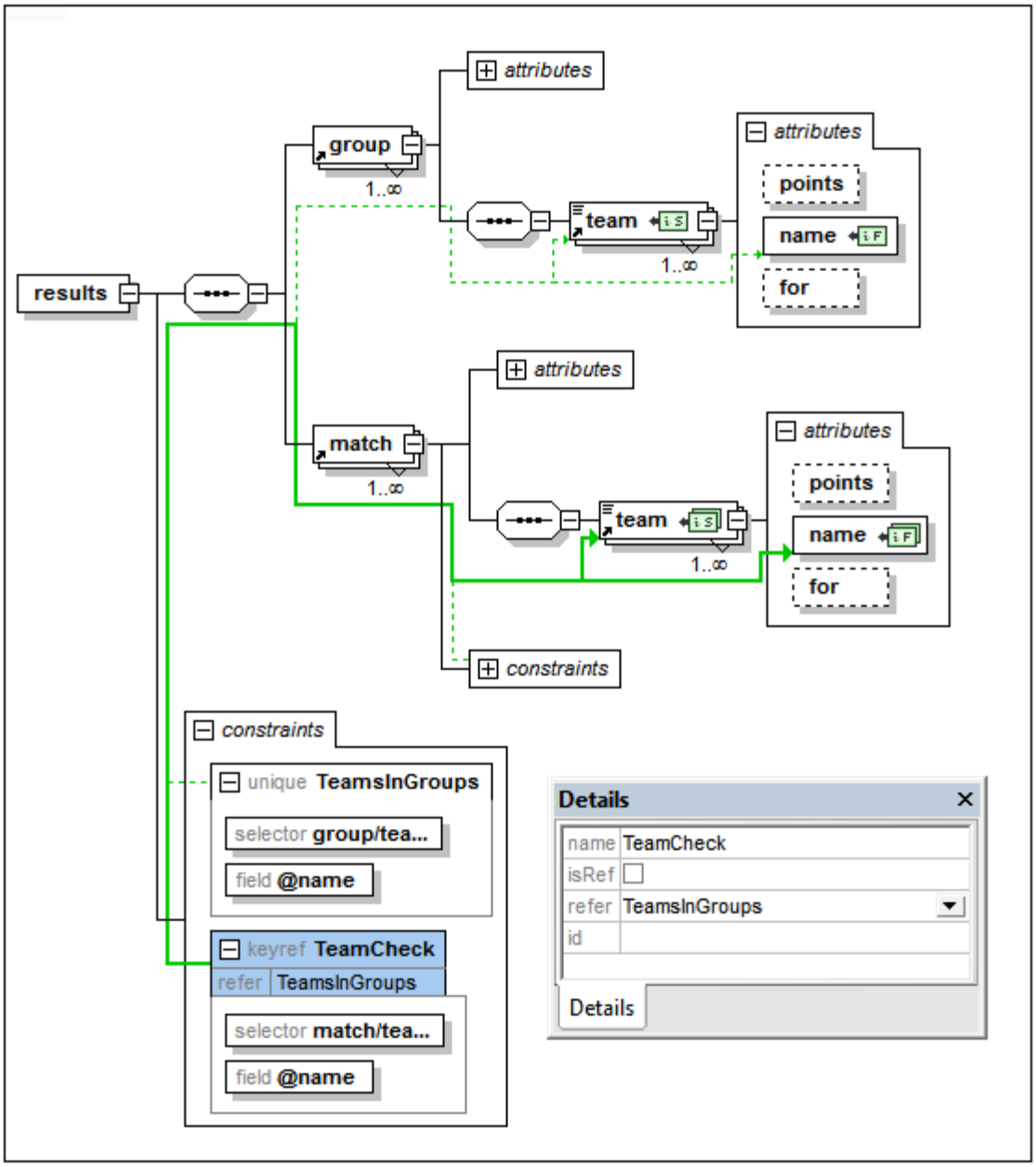
キー参照により、インスタンスドキュメント内にある値のセットがチェックされます。例えば上に示されるXML のテキストでは、キー参照を使うことにより同じグループに含まれているチーム同士で試合が行われるかをチェックすることができます。条件が満たされない場合、XMLドキュメントの検証に失敗します。

最初にユニーク制約またはキー制約を作成します。以下のスクリーンショットではTeamInGroups という名前のユニーク制約がresults 要素に対して作成されています。この制約により、group 要素内にある各 team にて、ユニークな @name 属性値が必要になります。



次に、match の子要素である team を選択し、@name の属性値がTeamInGroups により返された値に含まれるかどうかをチェックする、TeamCheck という名前のキー参照(keyref) を作成します。

以下のスクリーンショットでは、(水色でハイライトされている) このキー参照と、詳細入力ヘルパーの内容が示されています。選択された同一性制約との関係は緑色の実線で示されており、選択されていない同一性制約は緑色の破線により示されています。それぞれの同一性制約にてselector とfield のXPath 条件式にて選択されたノードが、それぞれ と アイコンにより示されます。ノードが縮退されると、関係を表す線が破線状態で表示されます。



xpathDefaultNamespace の使用

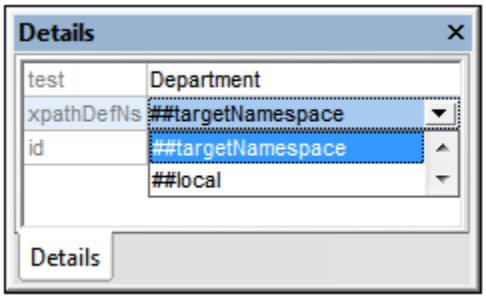
XML スキーマドキュメントで宣言されたデフォルト名前空間は、そのXML スキーマドキュメントのデフォルト名前空間です。スキーマドキュメント内の接頭辞のない要素名に対して適用されますが、スキーマドキュメント内の接頭辞のないXPath 条件式に対しては適用されません。

XSD 1.1 にて新たに追加された `xpathDefaultNamespace` 属性（ XSD 1.1 の新規機能）は、XPath 条件式にて接頭辞のない要素の名前空間を指定するために使用されます。XPath デフォルト名前空間は、それらが宣言されたXML スキーマ要素上でスコープされます。`xpathDefaultNamespace` 属性は、以下のXML Schema 1.1 要素内に記述することができます:

- `xs:schema`
- `xs:assert` と`xs:assertion`

- xs:alternative
- xs:selector とxs:field（同一性制約にて）

XSD 1.1 モードでは、xs:schema 上のxpathDefaultNamespace がスキーマ設定ダイアログ（**「スキーマ設計 | スキーマ設定」**）にてセットされます。上に示されるその他の要素の場合、**xpathDefaultNamespace 属性が詳細入力ヘルパーにてセットされます（以下のスクリーンショットを参照）**。xs:schema 上でXPath デフォルト名前空間を宣言することで、スキーマ全体に対するXPath デフォルト名前空間が宣言されます。この宣言は、xpathDefaultNamespace 属性が許されている属性でオーバーライドすることができます（*上のリストを参照ください*）。



Xs:schema 上でXPath デフォルト名前空間を宣言することで、スキーマ全体に対するXPath デフォルト名前空間が宣言されます。この宣言は、xpathDefaultNamespace 属性が許されている属性でオーバーライドすることができます（*上のリストを参照ください*）。

実際の名前空間を含める代わりにxpathDefaultNamespace 属性には、以下のうちのどれかを含めることができます:

- ##targetNamespace: スキーマのターゲット名前空間と同じ名前空間がXPath デフォルト名前空間として使用されます。
- ##defaultNamespace: スキーマの名前空間と同じ名前空間がXPath デフォルト名前空間として使用されます。
- ##local: XPath デフォルト名前空間は指定されません。

ドキュメントに対してXPath デフォルト名前空間が宣言されなかった場合、XPath 条件式における接頭辞なしの要素は名前空間を持たない要素として扱われます。

メモ XPath デフォルト名前空間宣言は属性に対して適用されません。

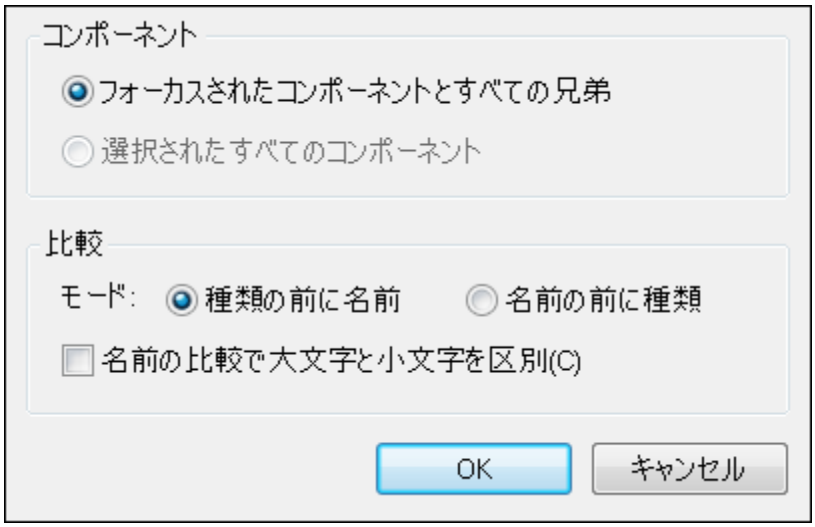
同一性制約のID

同一性制約とそのセレクター、フィールドに対してID を与えることができます。ID を割り当てるには必要なコンポーネントを選択し、詳細入力ヘルパーのID 行にてID を入力してください。

同一性制約の並べ替え

同一性制約タブのツールバーにある「**並べ替え**」アイコンをクリックすることで、同一性制約の並べ替えを行うことができます。コンポーネントの並べ替えダイアログでは、選択されたコンポーネントとその兄弟、または選択されたコンポーネントのセットを並べ替えの対象とすることができます。「Shift」キーを押下しながらクリックすることで範囲の選択を、「Ctrl」キーを押下しながらクリックすることで新たなコンポーネントを選択対象に含めることができます。

(C) 2015-2021 Altova GmbH



並べ替えの範囲を選択した後には、すべての対象をアルファベット順で並べ替えるのか（種類の前に名前）、または同一性制約の種類ごと（ユニーク制約、キー制約、キー参照の順）でアルファベット順に並べ替えるのかを選択することができます。

並べ替えの順序はスキーマのテキストに反映されます。

4.4.5　スキーマ ビューにおける入力ヘルパー

スキーマビューにて 3 つの入力ヘルパーが表示されます。このセクション以下にあるサブセクションでは、これら入力ヘルパーの詳細について説明されます:

- コンポーネント入力ヘルパー 201
- 詳細入力ヘルパー 205
- ファセット入力ヘルパー 207

表示される入力ヘルパーはスキーマ概要とコンテンツ モデルビューで変わらず、スキーマ コンポーネントの定義を視覚的に追加ならびに編集することができます。入力ヘルパーからコンポーネントをドラッグすることができるほか、デザイン上でコンポーネントを選択して、そのプロパティを入力ヘルパーで定義することができます。

4.4.5.1　コンポーネント

スキーマビューにおけるコンポーネント入力ヘルパーでは、3 つの操作を行うことができます:

- コンポーネントの種類と名前空間により、ツリー内にあるグローバルコンポーネントを管理する（以下のスクリーンショットを参照）: これにより、名前空間に従うかたちですべてのグローバルコンポーネントの概要が表示されます。
- グローバルコンポーネントのコンテンツ モデルビューへのナビゲーションを行う（コンポーネントにコンテンツ モデルが含まれている場合）: コンポーネントにコンテンツ モデルが含まれない場合、スキーマ概要にてコンポーネントがハイライトされます。他のスキーマからインクルードまたはインポートされたグローバルコンポーネントもコンポーネント入力ヘルパーに表示されます。
- スキーマドキュメント内で定義された同一性制約の概要を表示する。同一性制約タブの詳細については、「同一性制約」194 のセクションを参照ください。

メモ 表示されるのがXSD 1.0 なのか1.1 なのかは、選択されているXSD モード（XSD 1.0 または1.1）により決定されます。

グローバルタブ

グローバルタブでは、グローバルコンポーネントがツリー構造でグループ化され表示されます（以下のスクリーンショットを参照）。スキーマ内にある各グローバルコンポーネントの種類の数が、各種コンポーネントの隣に表示されます。

ツリー構造では、グローバルコンポーネントが以下の7 つの種類に分類されます。コンポーネントの種類がツリーに表示されるには、その種類のコンポーネントがスキーマ内部に最低 1 つ必要になります。

- 要素宣言（要素）
- モデルグループ（グループ）
- 複合型
- 単純型
- 属性宣言（属性）
- 属性グループ
- 記法

名前空間タブ

名前空間タブでは、名前空間とコンポーネントの種類により分類されたかたちでコンポーネントが表示されます（以下のスクリーンショットを参照）。

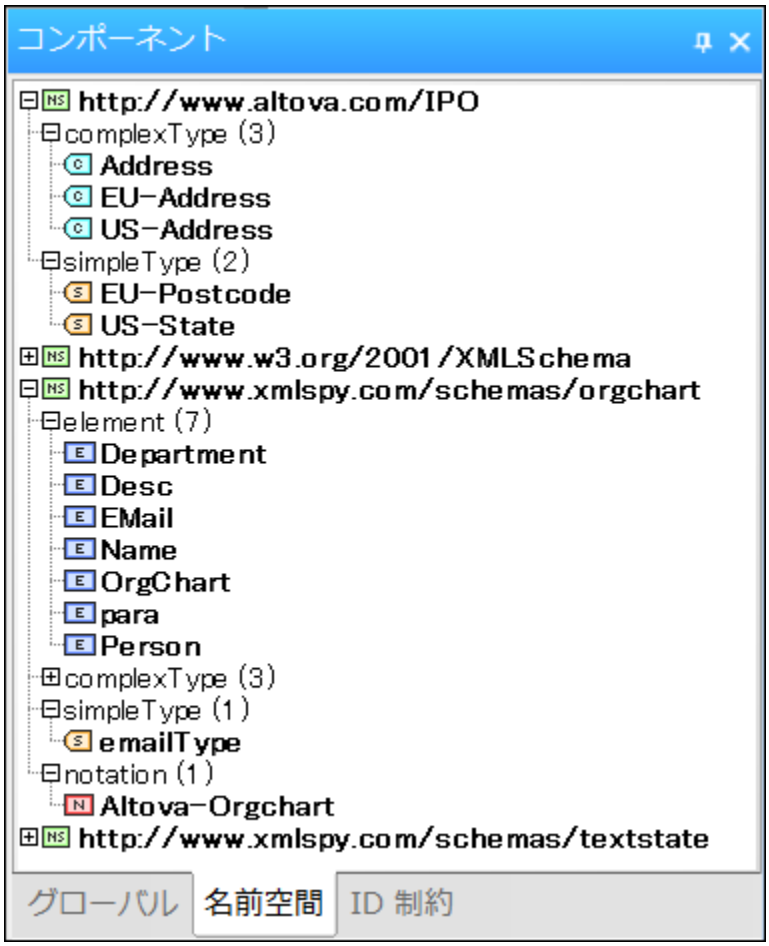
(C) 2015-2021 Altova GmbH

ツリーの表示では、グローバルコンポーネントが以下にある 7 つのグループに分類されます。コンポーネントの種類がツリーに表示されるには、その種類のコンポーネントがスキーマ内部に最低 1 つ必要になります。

- 要素宣言（要素）
- モデルグループ（グループ）
- 複合型
- 単純型
- 属性宣言（属性）
- 属性グループ
- 記法

グローバルと名前空間タブに表示されるコンポーネントのグループ

グローバルまたは名前空間タブにてコンポーネントのグループを展開すると、そのグループに含まれるすべてのコンポーネントが表示されます。これにより、ユーザー定義のコンポーネントへ簡単にアクセスすることができます。コンポーネント タブに表示されたコンポーネントをダブルクリックすると、その定義がメイン ウィンドウに表示されます。

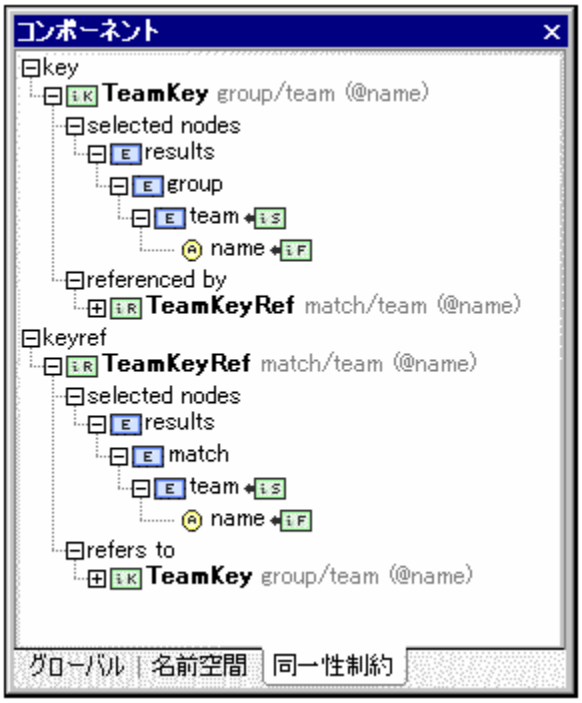


コンポーネントにコンテンツ モデルが含まれている場合（つまり、コンポーネントが要素、グループ、または複合型の場合）、コンポーネント名をダブルクリックすることで、コンポーネントのコンテンツ モデルが（メイン ウィンドウにある）コンテンツ モデルビューに表示されます。コンテンツ モデルを含まないコンポーネントがダブルクリックされた場合、そのコンポーネントが（メイン ウィンドウにある）スキーマ概要でハイライトされます。

注意: インクルードまたはインポートされたスキーマの中にコンポーネントが含まれる場合、インクルード/インポートされたスキーマが開かれ、コンポーネントのコンテンツ モデルがコンテンツ モデルビューに表示されるか、コンポーネントがスキーマ概要でハイライトされます。

同一性制約

コンポーネント入力ヘルパーの同一性制約タブでは、ドキュメントにある同一性制約の概要が表示されます（以下のスクリーンショットを参照）。このタブでは、同一性制約の種類（ユニーク（`unique`）、キー（`key`）、キー参照（`keyref`））により分類されたかたちで同一性制約が表示されており、展開/縮退可能なツリーにより同一性制約が表示されます。



太字で表示されているエントリーはその同一性制約が現在アクティブなスキーマ内部にあることを表しており、そうでないエントリーはサブスキーマ内に含まれることを表しています。同一性制約タブに表示されたエントリーをダブルクリックすることで、そのスキーマ コンポーネントがコンテンツ モデルビュー 194 にて表示されます。

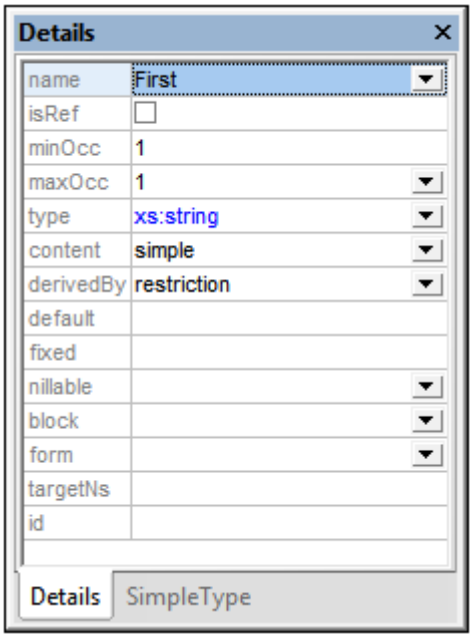
同一性制約タブにてアイテムが選択されている状態では、以下のコンテキスト メニュー コマンドを利用することができます:

- **図に表示**: スキーマ コンポーネントをコンテンツ モデルビュー 194 に表示します。
- **図にセレクター/フィールド ターゲットを表示**: 同一性制約のセレクター（selector）またはフィールド（field）にて指定されているスキーマ コンポーネントをコンテンツ モデルビュー 194 に表示します。複数のフィールドがある場合、目的のフィールドを選択するためのダイアログが表示されます。
- **同一性制約へ移動**: 同一性制約をスキーマ概要 194 に表示します。
- **すべての子を展開/縮退**: ツリーを展開/縮退します。

同一性制約タブの詳細については、「同一性制約」 194 のセクションを参照ください。

4.4.5.2 詳細

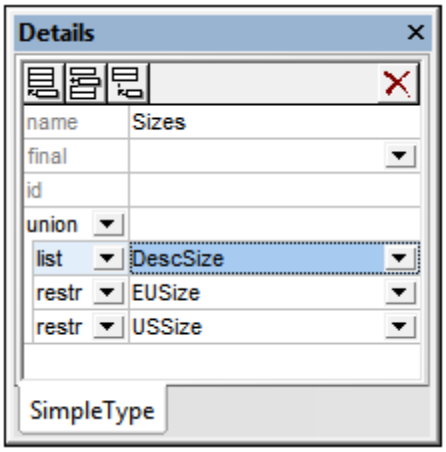
スキーマ ビューの詳細入力ヘルプ パーネルは、メイン ウィンドウにて現在選択されているコンポーネントやコンポジッターに関する編集可能な情報が表示されます。データベース拡張を含んでいるスキーマ ファイルを編集している場合、DB 拡張に関する情報を表示するためのタブも表示されます。



現在選択されているコンポーネントやコンポジッターのプロパティを変更するには、編集するフィールドをダブルクリックして、値を選択するか直接入力してください。編集するフィールドにてコンボボックスが利用できる場合、ドロップダウンリストから目的の値を選択してください。詳細入力ヘルパーにて行った変更は、直ちにデザインへ反映されます。

単純型の派生

詳細入力ヘルパーにて派生した単純型（restriction、list、union）を素早く正確に作成することができます。デザインにて単純型が選択されると、詳細入力ヘルパーに単純型タブが表示されます（以下のスクリーンショットを参照）。



単純型タブにある派生型のコンボボックスにて派生型（restriction、list、union）を選択し、右側にある対応するメンバーの型で利用可能な単純型から目的の単純型を選択してください。ツールバーにあるアイコンを使用することで、同じレベルに型を追加または挿入し、サブレベルに別の派生を追加し、派生型を削除することができます。型の定義に移動するには、「**型の定義**」をクリックします。この場合、ビルトインの単純型の場合、単純型の情報を含むメッセージボックスが表示されます。

(C) 2015-2021 Altova GmbH

4.4.5.3 ファセット

単純型の基底型（既存の単純型）に制約を課すことで新たな単純型（名前付き、または匿名型）が作成されます。このような制約は、基底型の値にファセットを追加することで達成されます。スキーマ ビューにあるファセット入力ヘルパーでは、単純型のファセットを簡単かつ視覚的に編集することができます（以下のスクリーンショットを参照）。利用可能なファセットは、以下のテーブルで示されるようにファセット入力ヘルパーのタブにて管理されます。

タブ	利用可能なファセット
ファセット 207	minInclusive、maxInclusive、minExclusive、maxExclusive、length, minLength、maxLength、totalDigits、fractionDigits、whiteSpace、explicitTimezone
パターン 208	pattern
列挙 208	enumeration
アサーション 209	assertion
サンプル 210	altova:exampleValues はサンプルであり、ファセットではありません。サンプルは XMLSpy にて XML スキーマからインスタンス XML ドキュメントを生成する際のサンプル値として使用されます。

以下に、これらタブの詳細について記述します。

デザイン内にある単純型の選択

単純型（名前付きまたは匿名型）は、以下のデザイン環境にて選択することができます:

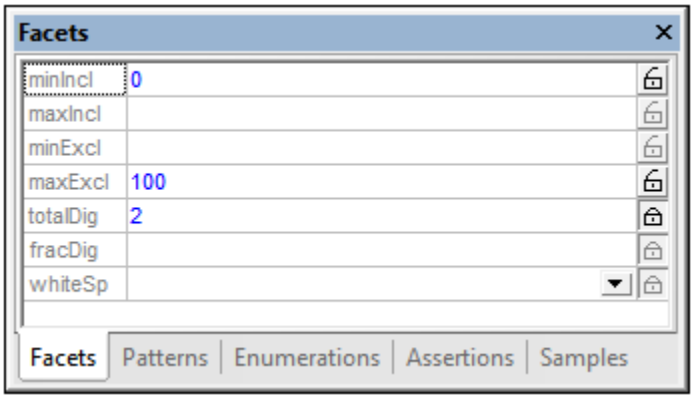
- スキーマ概要 156 にて: グローバルコンポーネントのリスト、またはグローバルコンポーネントのリスト以下にある属性タブから。
- コンテンツ モデル ビュー 168 にて: 図の中、またはデザイン下部にある属性タブから。

上記のデザイン環境にて単純型が選択されると、定義されているファセットがファセット入力ヘルパーに表示され、編集することができるようになります。

ファセット タブ

ファセット タブではデザインにて選択されている型に対して利用可能なファセットだけが表示されます。例えば制約を課されているのが xs:string 型の場合、totalDigits などの利用できないファセットは表示されません。

- 数値に関するファセット（minInclusive、maxInclusive、minExclusive、maxExclusive）は数値型と日付/時間型、そしてこれらの型から派生した型に対してだけ使用することができます。
- 長さに関するファセットは string を基底とした型、バイナリ型、そして anyURI に対してだけ使用することができます。
- totalDigits ファセットは xs:decimal ならびに整数型と、これらの型から派生したすべての型に対して使用することができます。fractionDigits ファセットは xs:decimal に対してのみ適用されます。

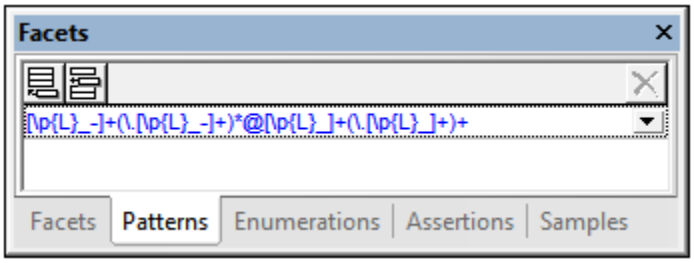


値を入力するには、(それが存在する場合には) コンボ ボックスのドロップダウン リストから値を選択、または値フィールドをダブルクリックして値を入力してください。無効な値や他の値と相反する値入力された場合、その値が赤色で表示され、妥当な値は青色で表示されます。例えば、`minInclusive` ファセットと`maxInclusive` ファセットは同時に定義することができないため、これらのうち後に入力された値が赤色で表示されます。

固定ファセット (fixed="true" の属性を持つファセット) を指定するには、ファセットの右側にある鍵のシンボルをクリックして、シンボルを閉じた鍵に変更してください。上のスクリーンショットでは、`totalDigits` ファセットが固定ファセットとして定義されています。固定することのできるファセットの数に制限はありません。ファセットの固定を解除するには、閉じた鍵のシンボルをクリックしてください。

パターン タブ

パターン タブでは、1 つ以上の`pattern` ファセットを制約に追加することができます。`pattern` ファセットのパターンは正規表現の構文により指定されます。以下のスクリーンショットでは、電子メールアドレスのパターンが示されます。



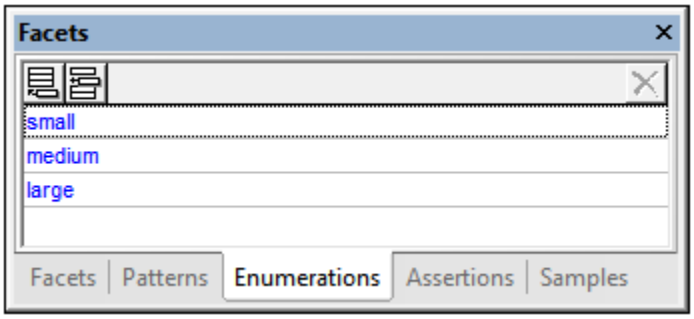
複数の`pattern` ファセットが指定されている場合、XML インスタンスの値が指定されたパターンの 1 つにマッチする必要があります。例えば、米国と欧州の郵便番号にマッチするパターンが与えられている場合、XML インスタンスの値を、そのうち 1 つに対してマッチさせる必要があります。

左上にある「**追加**」または「**挿入**」アイコンをクリックして、目的のパターンを定義する正規表現を入力することで、`pattern` アセットを追加することができます。パターンを選択して、右上にある「**削除**」アイコンをクリックすることで、そのパターンを削除することができます。

列挙タブ

列挙タブでは、`enumeration` ファセットを制約に加えることができます (以下のスクリーンショットを参照)。各 `enumeration` ファセットにより、その型に対して妥当となる値が指定されます。`enumeration` ファセットのセットにより、許されている値のセットを指定することができます。以下のスクリーンショットでは、`enumeration` ファセットにより許されている文字列のセットを確認することができます。

(C) 2015-2021 Altova GmbH



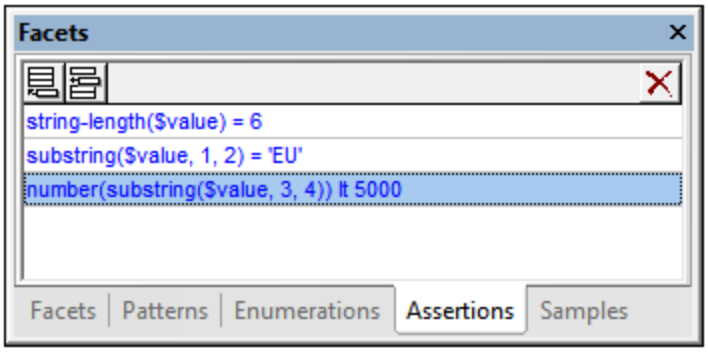
左上にある「**追加**」 または「**挿入**」アイコンをクリックして enumeration の値を入力することで、enumeration ファセットを追加することができます。enumeration ファセットを選択し、右上にある「**削除**」アイコンをクリックすることで、その enumearation ファセットを削除することができます。

アサーション タブ

アサーションは XSD 1.1 にて新たに追加された機能で、XSD 1.1 モード 153 でのみ表示されます。ファセット入力ヘルパーのアサーション タブにて定義されたアサーション ファセットは、**単純型に対するアサーション**です（複合型に対するアサーションは、ファセット入力ヘルパーではなくスキーマ概要やコンテンツ モデルビューにて定義ならびに編集 191 されます）。

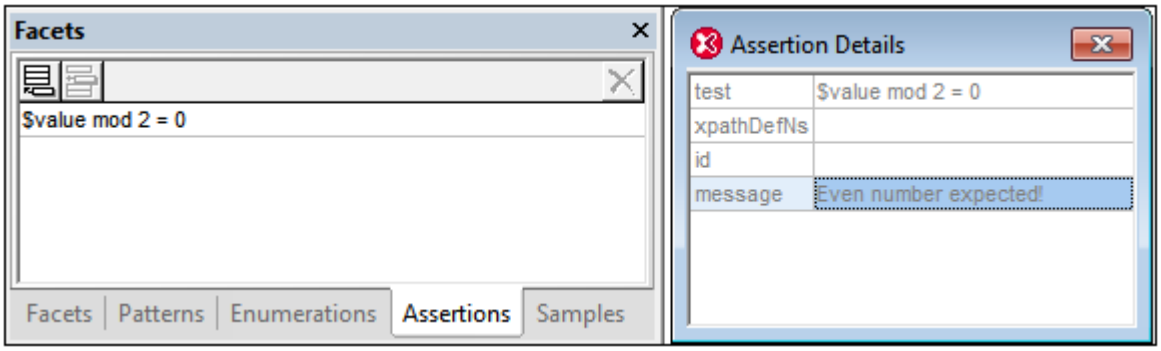
単純型（要素または単純コンテンツの属性）がデザインで選択されている際にアサーション タブへ切り替え、左上にある「**追加**」または「**挿入**」アイコンをクリックし、assertion を定義するのに使われる XPath 2.0 条件式を入力することで、アサーションの指定を行うことができます。XPath 条件式では、$value という特別な変数を使用することで、単純型の値を参照することができます（1 つの値以外に評価する要素はないため、通常使用される self::node() パス（またはその省略形であるピリオド '.'）を XPath 条件式内部で使用することはできません）。

例えば、string-length($value) = 6 という XPath 条件式により、単純型に含まれている値が 6 文字から構成されるかをチェックすることができます。インスタンス ドキュメント内に含まれている要素や属性の値が 6 文字から構成される場合、アサーションに従った妥当な値ということになります。



メモ　XPath 条件式における構文エラーは赤色により示されます。しかしデータ型はランタイムにて決定されるため、型に関するエラーは XPath 条件式の入力中に報告されません。そのため期待されたとおりに型の構築を行う必要があります。例えば上のスクリーンショットにある 3 番目の条件式では、数値の比較を行う前に、文字列の値（アサーションが xs:string 単純型にて定義されたと仮定した場合）を数値型へ変換しています。

上のスクリーンショットの例にあるように、複数のアサーションを同じ単純型へ指定することもできます。この場合、指定されているすべてのアサーションに要素または属性の値をマッチさせる必要があります。上のスクリーンショットにあるアサーションでは、インスタンス ドキュメントの値が 6 文字から構成され、EU という文字から開始され、最後の 4 文字を 0000 から 4999 までの数値に収める必要があります。



アサーションの詳細がこの定義と共に提供される場合とても役に立ちます。XML インスタンスドキュメントが検証される際、アサーションが満たされていない場合、適切な メッセージが表示されます。XML スキーマ 仕様は、このようなメッセージのために対応をしないため、XMLSpy は、アサーションの定義と共にXML インスタンスドキュメントの検証内で使用される `http://www.altova.com/ja/xml-schema-extensions` (または他の名前空間) 内でメッセージ を許可します。例えば

```
<xs:assert test="count(//MyNode) ge 1" altova:message="There must be at least one
MyNode element"/> or
<xs:assertion test="count(//MyNode) ge 1" altova:message="There must be at least one
MyNode element"/>
```

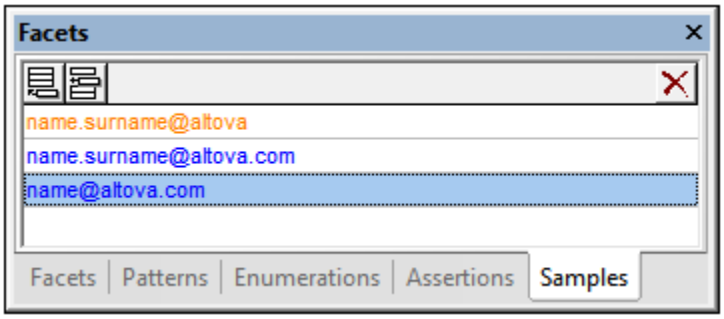
アサーション内で指定された制約が満たされない場合、XMLSpy の検証エンジンは、検証-エラーメッセージと共にヒントとしてのアサーションと関連したメッセージを表示します。バリデーターは、`assert/@message` 属性の値または、`message` 属性が存在する名前空間に関わらず、`assert/@message` 属性 を報告します。ですが、スキーマビュー内では、xml-schema-extensions 名前空間にある `message` 属性 のみを編集することができます。他の名前空間内の`message` 属性 を編集するには、テキストビューを使用します。

詳細に関しては、アサーションメッセージ 212 を参照してください。

メモ 可能な場合はアサーションよりもその他の種類のファセット使用する方が推奨されます。例えば、上のスクリーンショットにある最初のアサーションにより指定された制約は、(ファセット タブにある) `length` ファセットにより指定された方が良いでしょう。

サンプル タブ

サンプル タブでは、XML スキーマから XML インスタンス ファイルを生成する際のサンプル値として使用される値を指定することができます (以下のスクリーンショットを参照)。XML インスタンス ファイルの生成は、メニューオプションの「DTD/スキーマ | サンプル XML ファイルの生成」から行うことができます。サンプル値が無効の場合、警告としてサンプル値がオレンジ色で表示されます。以下のスクリーンショットでは最初のサンプルが電子メールの`pattern` ファセットにマッチしないため、無効な値となります (上記の パターン タブを参照ください)。



メモ ツールバーにある「**検証の警告を表示**」アイコン をクリックすることで、無効なサンプル値の警告を確認することができます。無効なサンプル値によりXSD ファイルの検証に失敗することはありません。

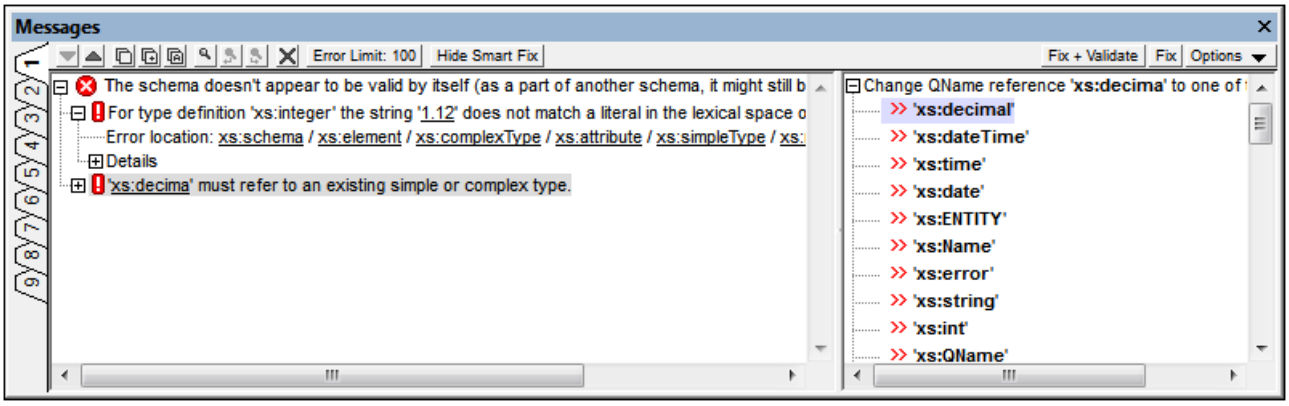
左上にある「**追加**」または「**挿入**」アイコンをクリックしてhttp://www.altova.com/ja/xml-schema-extensions内のサンプル値を入力することで、`altova:example` へサンプル値が入力されます。サンプル値を選択し、右上にある「**削除**」アイコンをクリックすることで、`altova:example` の値が削除されます。

4.4.6 検証とスマートフィックス

XML スキーマドキュメントの正確性の確認のために検証を行うことができます。これはメニューコマンド「**XML | XML の検証 (F8)**」をクリックして行います。

ドキュメントが有効な場合、この効果についてのメッセージがメッセージウィンドウに表示されます。

ドキュメントが無効な場合、メッセージウィンドウは、2 つのペインの表示をするように変更されます（*下のスクリーンショット参照*）。左側のペイン（エラーペイン）は、最初の x エラーまたはすべてのエラーをリストします。右側のペインは、スマートフィックスペインです。このペインは左側のペインで選択されたえらのためのフィックスの選択肢のリストを含みます。下のスクリーンショットの例では、エラーペイン内の2番目のエラーの選択は、右側のスマートフィックスペインにこのエラーのフィックスの選択肢の表示をします。フィックスの1つを選択してを「**フィックス+検証**」または**フィックス**選択すると、ドキュメント内のエラーはこの特定のフィックスと共に修正されます。



エラーペイン

ウィンドウのツールバーには、次の機能があります：

- 「**上へ**」および「**下へ**」矢印を使用して、エラーをスクロールします。
- メッセージ、またはメッセージとその子孫、またはすべてのメッセージをクリップボードにコピーします。
- 検索を使用して、用語を検索します。複数のエラーが報告されている場合、次を検索、および前を検索 機能は、役に立ちます。
- エラーペイン空すべてのエラーをクリアします。
- 検索され表示されるエラー数の制限を設定します（`1` から `999`）。デフォルトの値は `100` です。ボタンをクリックして制限を編集します。
- 表示/非表示 スマートフィックスペイン. スマートフィックスペインが非表示の場合、「**スマートフィックスを表示**」ボタンがツールバーに表示されます。クリックするとスマートフィックスペインが表示され、ボタンが「**スマートフィックスを非表示**」に変更されます。「**表示/スマートフィックスを非表示**」ボタンが、有効化されていない場合、スマートフィックスを使用することができません。

スマートフィックスペイン

ウィンドウのツールバーで次の機能を使用することができます:

- 「フィックス+ 検証」 ボタンは、選択されたエラーを選択された スマート フィックスで修正します。そしてドキュメントを再検証します。他のエラーは、エラーペインで再度開かれます。
- フィックス ボタンをクリックすると、エラーはフィックスされますが、再検証は行われません。
- オプション ボタンは、スマート フィックスをダブルクリックすることにより作動する振る舞いの選択を含むリストをドロップします: ダブルクリックにより「フィックス+ 検証」 または「フィックス」 を実行するかどうか。

4.4.7 アサーションメッセージ

XML Schema 1.1 内では、 複合型 (`xs:assert` 要素を使用して) および 単純型 (`xs:assertion` 要素を使用して) のために アサーションは、定義されることができます。

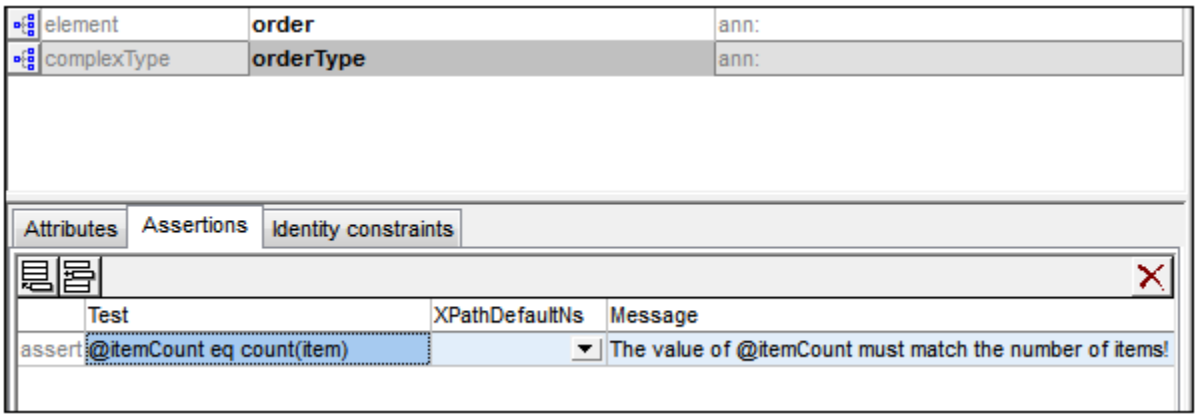
アサーションの詳細がこの定義と共に提供される場合とても役に立ちます。XML インスタンス ドキュメント が検証される際、アサーションが満たされていない場合、適切な メッセージが表示されます。XML スキーマ 仕様は、このようなメッセージのために対応をしないため、 XMLSpy は、アサーションの定義と共にXML インスタンス ドキュメント の検証内で使用される `http://www.altova.com/ja/xml-schema-extensions` (または他の名前空間) 内でメッセージ を許可します。例えば

```
<xs:assert test="count(//MyNode) ge 1" altova:message="There must be at least one
MyNode element"/> or
<xs:assertion test="count(//MyNode) ge 1" altova:message="There must be at least one
MyNode element"/>
```

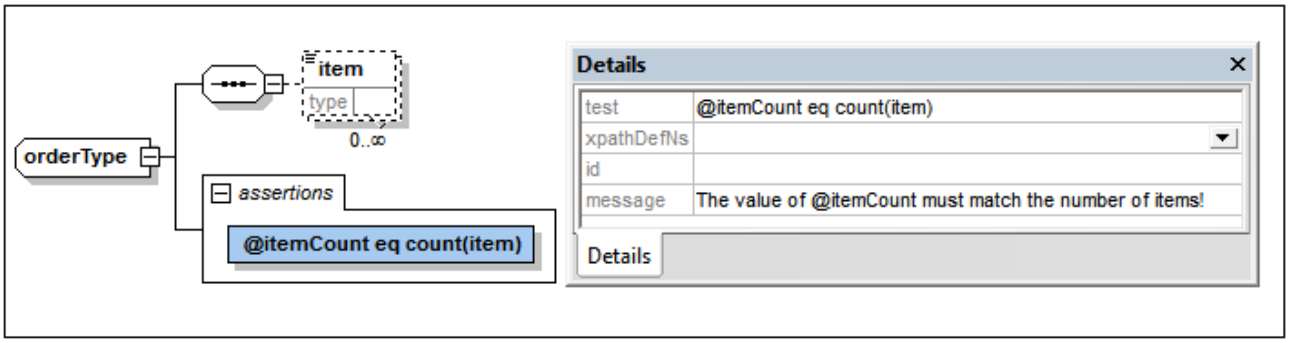
アサーション内で指定された制約が満たされない場合、XMLSpy の検証エンジンは、検証-エラーメッセージと共にヒントとしてのアサーションと関連したメッセージを表示します。バリデーターは、`assert/@message` 属性の値または、`message` 属性が存在する名前空間に関わらず、`assert/@message` 属性 を報告します。ですが、スキーマビュー内では、xml-schema-extensions 名前空間にある `message` 属性 のみを編集することができます。他の名前空間内の`message` 属性 を編集するには、テキストビューを使用します。

xs:assert メッセージの編集

スキーマビュー内では、(複合型のための) xs:assert 要素 を作成して、属性/アサーションID 制約 (AAIDC) ペイン 191 または関連する複合型の詳細入力ヘルパー 191 内で編集することができます。下のスクリーンショットは、複合型 `orderType` のためのアサーションを表示します。 (この場合は、`xs:assert`) は、(スキーマ概要の) アサーション タブ でアサーション メッセージと共に定義されています。

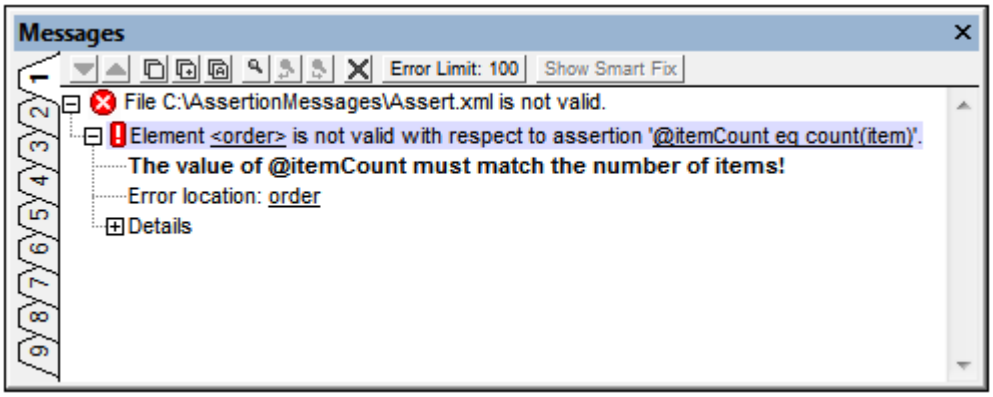


ダイアグラム内でアサーションを表示する 173 オプション が選択された場合、複合型 のアサーションが作成され、コンテンツモデルビュー内で編集されます。アサーション メッセージを追加または編集するには、アサーションを選択して、詳細入力ヘルパー内のアサーションメッセージに入力してください (*下のスクリーンショット参照*)。



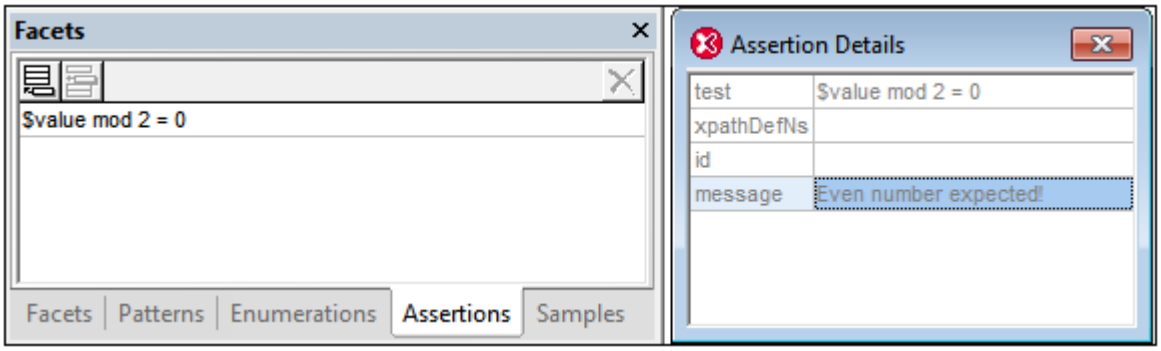
この方法で作成されるアサーションメッセージは、Altova xml-schema-extension 名前空間 http://www.altova.com/ja/xml-schema-extensions 内にあります。 XML スキーマドキュメント内に最初のアサーションメッセージを、AAIDC ペイン 191 または詳細入力ヘルパー 191 を介して追加すると、Altova xml-schema-extension 名前空間が自動的に xs:schema 要素に対して宣言されます。

XML ファイルが検証されてアサーションテストが満たされない場合、アサーションのために定義された、エラーメッセージと共に表示されます (*下のスクリーンショット参照*)。

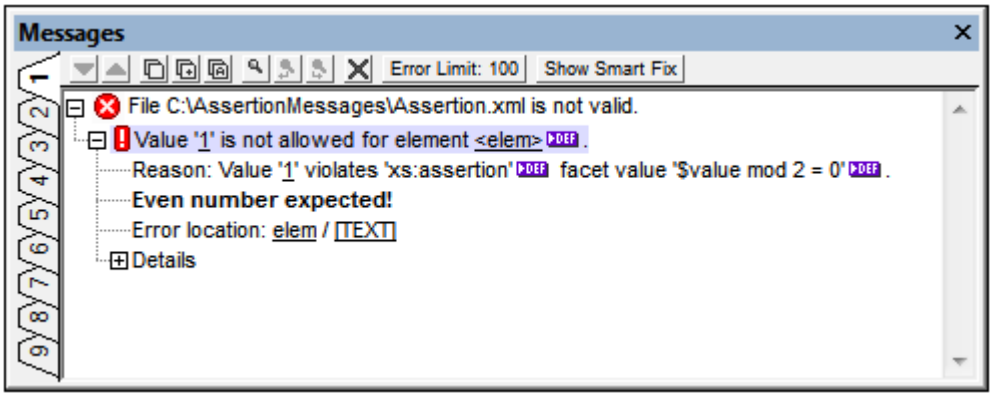


xs:assertion メッセージの編集

スキーマビューの xs:assertion 要素 (単純型) は、関連する単純型のファセット 入力ヘルパー 207 内で作成および編集されることができます。アサーションメッセージを編集するには、ファセット 入力ヘルパー (*下のスクリーンショット参照*) 内のアサーションを右クリックして、ポップアップするメニューの「**詳細**」をクリックし、アサーション詳細モーダルウィンドウのメッセージを編集します (*下のスクリーンショット参照*)。メモ この方法で作成される、アサーションメッセージは、Altova xml-schema-extension 名前空間 http://www.altova.com/ja/xml-schema-extensions 内にあります。アサーション詳細モーダルウィンドウを介して、XML スキーマドキュメント内のメッセージに最初のアサーションを追加する際には、名前空間は、自動的に xs:schema 要素 に対して宣言されます。



XML ファイルが検証されて、アサーションテストが満たされない場合、アサーションのために定義されたメッセージがエラーメッセージと共に表示されます（*下のスクリーンショット参照*）。

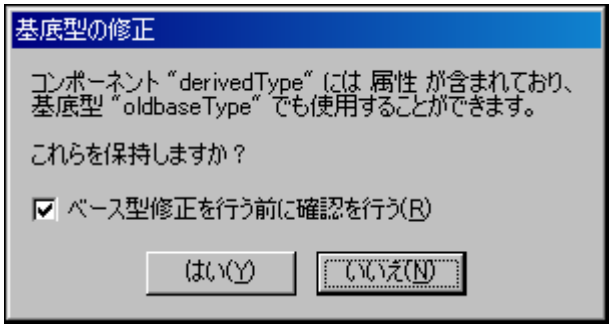


4.4.8 基底型の修正

派生した型の基底型がスキーマビューで変更されると、派生型にあるコンテンツ、属性、ファセット、サンプル値を、以下のいずれかの方法で処理することができます:

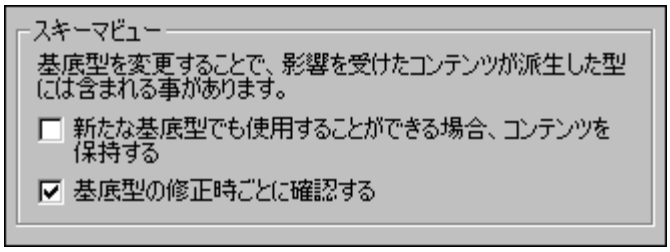
- 新たな基底型でも使用できる場合、これらの値を保持する。
- 新たな基底型でも使用できるかどうかにかかわらず、これらの値を自動的に削除する。

コンテンツ、属性、ファセット、またはサンプル値が含まれる派生型の基底型を切り替えると、基底型の修正ダイアログが表示されます（以下のスクリーンショットを参照）。



「基底型の修正を行う前に確認」チェック ボックスを無効にした場合、オプション ダイアログのビューのセクション（「ツール | オプション | ビュー[1025]」）にてこの設定を再度有効にすることができる旨のメッセージが表示されます。

オプション ダイアログのビューのセクション（「ツール | オプション | ビュー[1025]」）にあるスキーマ ビュー ペインでは、コンテンツを保持するか、または基底型の修正を行うたびにユーザーへの確認を行うかを指定することができます（以下のスクリーンショットを参照）。



コンテンツを保持して、確認を行うチェック ボックスを有効にすることで、これらの動作がデフォルト のオプションになります。

4.4.9　スマート制約

複合型に対して制約を課す場合、基底型に含まれるコンテンツ モデルの一部が派生型にて書き換えられることになります。派生型の編集を行いながらその基底型のコンテンツ モデルを正確に把握するのは大変な作業です。

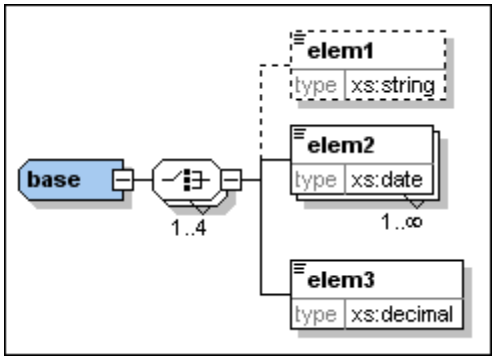
スマート制約により、（基底型と派生型の）2 つのコンテンツ モデルが派生した型のコンテンツ モデルに表示されます。派生した複合型では、基底複合型にあるすべての項目と、それらが派生した型とどのように関連付けられているのかを確認することができます。更にスマート制約では、基底型のコンテンツ モデルへ課すことのできるすべての制約が表示されます。これで派生した型を簡単かつ正確に制約することが可能になります。

以下の操作によりスマート制約を有効にすることができます:

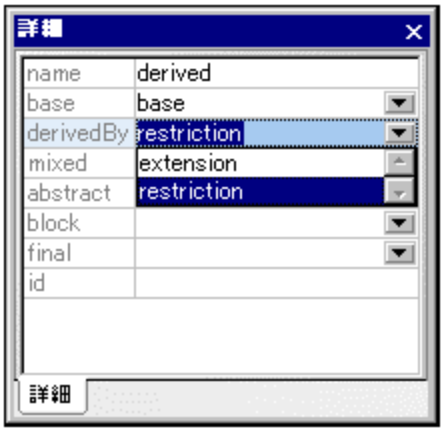
- スキーマ デザイン ツールバーにあるスマート制約アイコン をクリックします。

以下のサンプルはスマート制約の機能を説明しています。

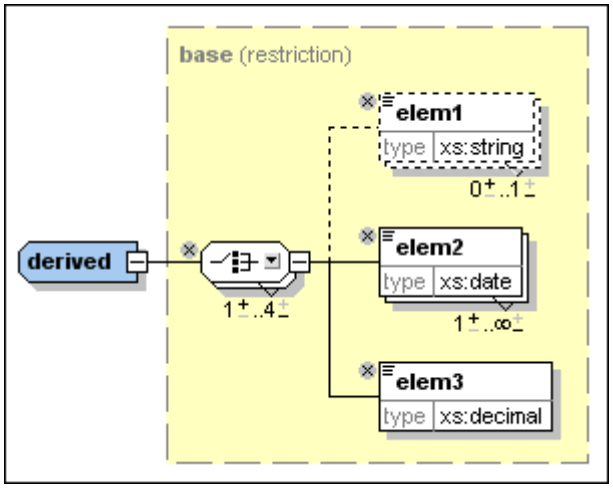
この例では以下に示される複合型が基底型として使用されます



上に示される基底型のbase から、以下のように複合型のderived を派生させることができます:
1.スキーマにて derived という名前の複合型を新たに作成します。
2.詳細入力ヘルパーにて base ドロップダウン リストから base を選択し、derivedBy ドロップダウン リストから restriction を選択します。

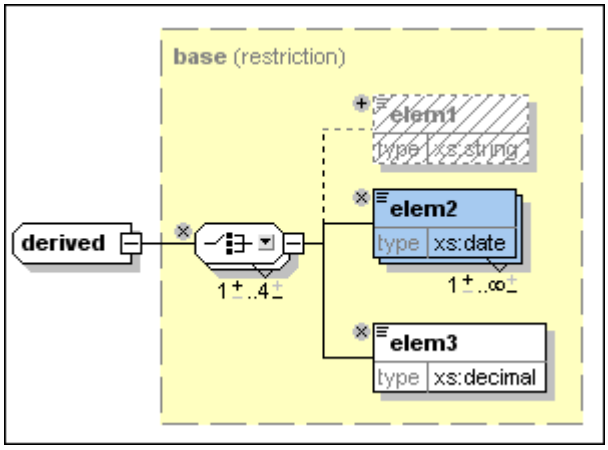


スマート制約が有効になっている状態で、派生型が以下のように表示されます

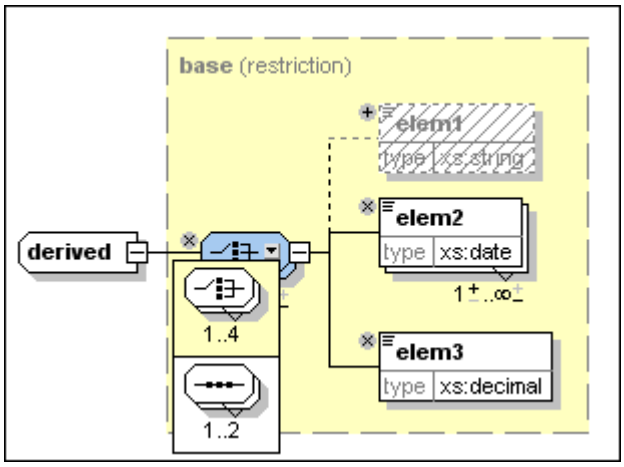


この例で派生型の制約に使用されるコントロールを以下に示します:

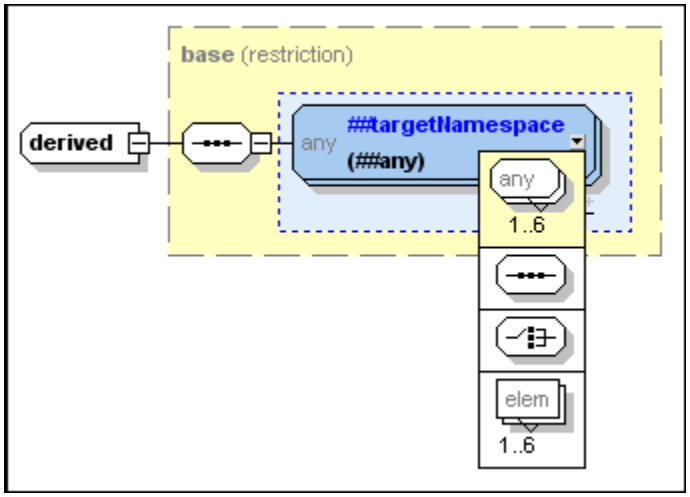
- このアイコン により、基底型にある要素を削除することができます。ここでは elem1 が削除されます。要素を追加するには、アイコン をクリックしてください。



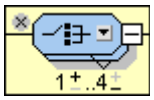
- 選択コンポジッターに表示された矢印 をクリックすることで、以下のスクリーンショットに示されるリストが表示されます。選択モデル グループをシーケンス モデル グループへ変更することができます。

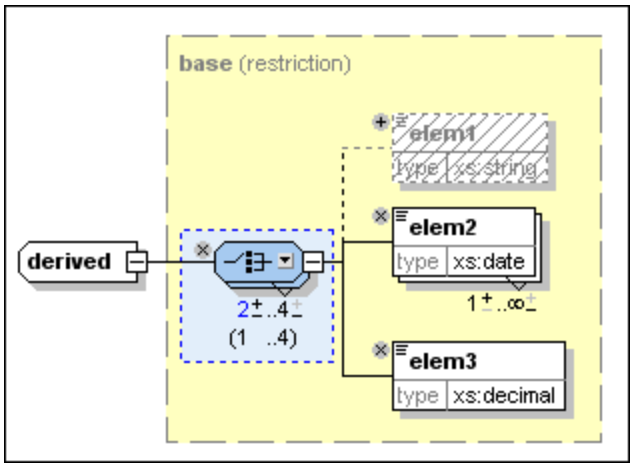


同様に、ワイルドカードを変更することもできます



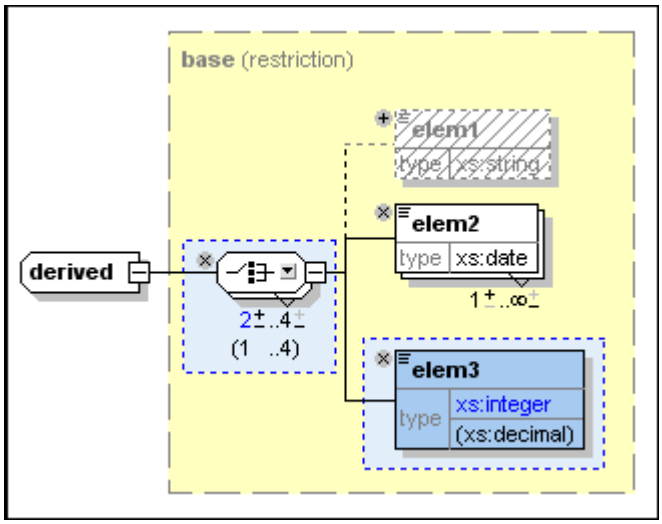
どの項目をどの項目により置き換えることができるかの詳細については、XML スキーマの仕様 (英文) を参照ください。

- モデル グループの出現回数は、  にある 1 の隣にあるプラス記号により最低出現回数を増やし、4 の隣にあるマイナス記号により最大出現回数を減らすことができます。基底型にて定義された出現の回数が特定の数 (例えば 4) ではなく範囲の場合 (例えば 2-5) の場合にこれらのコントロールが表示されます。出現の範囲が間違っていてもこれらのコントロールは表示されます。

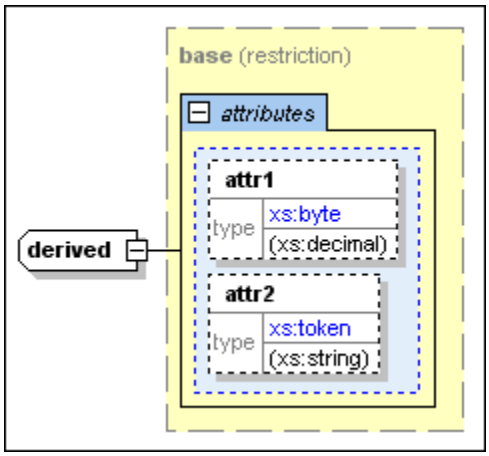


上のスクリーンショットでは、要素の最低出現回数が 2 に変更されています。モデル グループの背景が青色に変更され、基底複合型にあるモデル グループとは違うものであることを表している点に注目してください。基底型にて定義された出現範囲も表示されます。

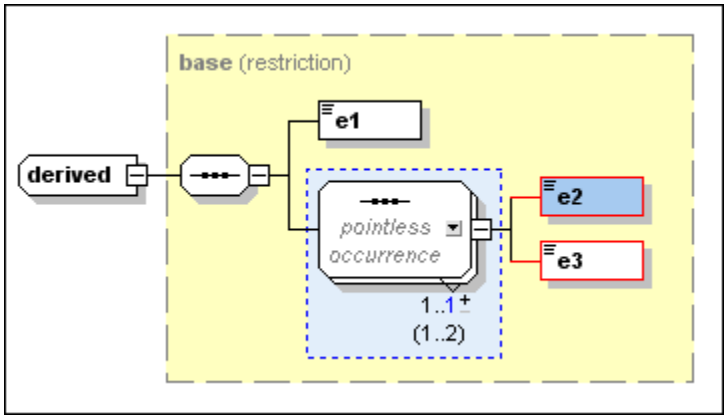
- それが XML スキーマの仕様で定義された基底型の制約に対して妥当な変換である場合、属性または要素のデータ型を変更することができます。例えば、derived 複合型にある elem3 データ型を decimal から integer へ変更することができます。これにより、要素の背景が青色に変わり、基底型のデータ型とは違っている点が示され、基底型で定義されている要素のデータ型がかっこの中に表示されます:



派生した複合型にて値のデータ型が制限された属性を以下のスクリーンショットに示します



- スマート制約はコンテンツ モデルで**意味のない出現**の警告を行います。意味のない出現とは、例えばコンテンツ モデルで表示されている、実際には必要ないシーケンスのことです。以下の図に意味のない出現の例を示します:



メモ: 意味のない出現は、コンテンツ モデルがエラーを含んでいる時にだけ表示されます。コンテンツ モデルが意味のない出現を含んでいながら妥当である場合もあり、その場合は混乱を避けるために意味のない出現は明示的に表示されません。

意味をなさない表示の詳細については、XML スキーマの仕様を参照ください。

4.4.10 xml:base、xml:id、xml:lang、xml:space

名前空間 `http://www.w3.org/XML/1998/namespace` はXML 名前区間の仕様により与えられたもので、`xml:` 接頭辞に紐付けられています。これにより、`xml:` 接頭辞とともにこの名前空間が使用されることになります。この名前空間では4 つの属性が定義されており、任意のXML ドキュメント（スキーマかインスタンスかを問わず）にあるすべてのXML 要素にて使用することができます:

- `xml:base`: 要素のベースURI をセットするのに使用
- `xml:id`: 要素のユニークなID を指定するのに使用
- `xml:lang`: その要素内にあるコンテンツが記述されている言語を識別するために使用
- `xml:space`: 要素内にある空白スペースをどのように処理するかを指定するために使用

スキーマビューにて、XML スキーマドキュメントへこのXML 名前空間がインポートされると、これら4 つの xml: 属性をスキーマ内にあるすべての要素にて使用することができます。

以下の操作により、これら属性のうち 1 つを要素にて宣言することができます:

1. そのスキーマドキュメントに対するXML Namespace 名前空間を宣言し、名前空間を xml: 接頭辞に割り当てます。4 つある xml: 属性のうちのどれかがドキュメントにて使用されると、その名前が正しい名前空間を含めるように展開されます。
2. XML Namespaces 名前空間をインポートします。XMLSpy のバリデーターが名前空間を認識し、4 つの xml: 属性がスキーマ内部から参照可能なグローバル属性で利用できるようになります。
3. 目的の xml: 属性を要素の子として挿入します。インポートされたグローバル属性への参照として属性は宣言されます。

XML Namespaces 名前空間の宣言

そのスキーマに対して宣言されているすべてのスキーマを確認ならびに編集することができるスキーマ設定ダイアログでは、XML Namespaces 名前空間（http://www.w3.org/XML/1998）を入力することができます。名前空間は xml: 接頭辞へ関連付ける必要があります（xml: 接頭辞とともに、テキスト ビューで表示された xs:schema 要素にて名前空間を宣言することもできます）。

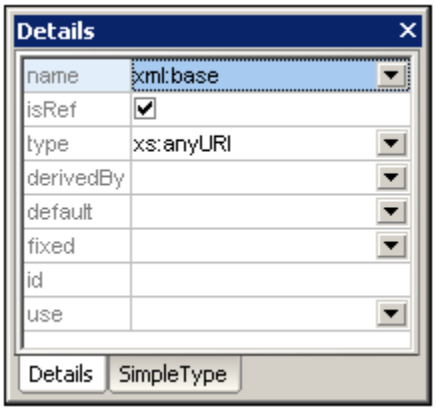
XML Namespaces 名前空間のインポート

スキーマ概要にて XML Namespaces 名前空間に対するグローバル要素宣言を作成します。スキーマ概要ウィンドウの上部にある挿入 または追加 アイコンをクリックし、メニューから「インポート」を選択します。XML Namespaces 名前空間をインポートする名前空間として入力します。テキスト ビューではインポート宣言が以下のように表示されているはずです:

```
<xs:import namespace="http://www.w3.org/XML/1998/namespace"
schemaLocation="http://www.w3.org/XML/1998/namespace"/>.
```

xml: 属性の追加

スキーマ概要にて xml: 属性を追加する要素を選択し、その要素に対して属性を追加します。詳細ウィンドウにて name コンボボックスにある矢印ボタンをクリックし、目的の xml: 属性（例えば xml:base）を選択します（以下のスクリーンショットを参照）。グローバル属性を参照するか尋ねられた場合、「はい」を選択します。属性が参照として追加されます。



XInclude と xml:base

XInclude の include 要素がインクルード要素の href 属性にて指定された XML ファイルにより置き換えられた場合、解析された XML ドキュメントの最上レベルにある要素に xml:base 属性が与えられます。この XML ドキュメントを検証する必要がある場合、xml:base 属性を関連する要素に対して定義する必要があります。

4.4.11 戻ると進む: 位置の移動

スキーマビューの「**戻ると進む**」コマンドにより、スキーマビューで直前に閲覧した位置に戻ることができます。スキーマビューにてスキーマコンポーネント間を移動しながら、既に見たコンポーネントを再度閲覧するのに便利な機能です。ツールバーにある「**戻る**」ボタンを一度クリックすることで、直前に閲覧した位置に移動します。「**戻る**」ボタンを繰り返し押すことで、既に閲覧したポジション（最大 500）まで移動することができます。既に閲覧した位置に移動した後、「**進む**」ボタンをクリックすることで先の位置に移動することができます。

これら 2 つのショートカット キーを以下に示します:

- **戻る**: Alt + 左矢印
- **進む**: Alt + 右矢印

戻る/進むと元に戻す/やり直し

戻ると進むコマンドは**元に戻す**（「Ctrl」 + Z）や**やり直し**（「Ctrl」 + Y）とお違うものだということに注意してください。「**戻る**」コマンドをクリックすることで、最後に見た状態のまま直前に閲覧したコンポーネントに戻ります。元に戻すコマンドを一度選択すると、最後に行った変更を（その変更がどこで行われたかに関わらず）取り消します。

追加メモ

以下の点に注意してください:

- 「**戻る**」ボタンにより、最大 500 回まで最後に閲覧した場所に移動できます。
- 戻る/進む機能はスキーマ全体に対して有効になります。スキーマが閉じられ、他のビューで開かれていたとしても、スキーマビューにて再度開かれます。
- 前に閲覧したコンポーネントが削除されていた場合、コンポーネントを閲覧することはできません。そのようなコンポーネントが前の位置の一部であった場合、削除されていないコンポーネントとともに以前の画面が表示されます。削除されたコンポーネントが以前閲覧した位置のすべてを含んでいた場合、その位置を表示することはできず、「**戻る**」ボタンをクリックしても、削除されたコンポーネントのさらに 1 つ前の位置に移動します。

(C) 2015-2021 Altova GmbH

4.5 Authentic ビュー

Authentic View には、ウィンドウの上部にメニューバーとツールバーがあり、残りのインターフェイスは以下の3つの部分から構成されています: プロジェクトウィンドウ、メインウィンドウ、入力ヘルパー。これらの部分は以下に表示されています。

Authentic ビューインターフェイス[415] の詳細は **Authentic**[401] | Authentic ビューインターフェイス[415] のセクションで説明されています。

4.6 ブラウザービュー

ブラウザービューは、通常以下を閲覧するために使用されます:

- 関連したXSLT ファイルをもつXML ファイル。ブラウザービューに切り替えると、XML ファイルは、関連したXSLT スタイルシートを使用して素早く変換され、結果は直接ブラウザービューに表示されます。
- HTML ファイルは、HTML として直接作成されるか、またはXML ファイルのXSLT 変換 を介して作成されます。

ブラウザービュー内のXML およびHTML ファイルを閲覧するには、**ブラウザー** タブをクリックします。

Microsoft Internet Explorer およびXSLT に関する注意点

ブラウザービューは、またMicrosoft Internet Explorer 5.0 または以降を必要とします。XSLT スタイルシートにより変換されたXML ファイルを閲覧するためにブラウザービューを使用する場合は、XSLT 1.0 標準を完全にサポートするXML パーサであるMSXML 3.0 をサポートするInternet Explorer 6.0 または以降を使用することが奨励されます。MSXML 4.0.のインストールを行う場合は、(IE 5 内のXSLT へのサポートは、公式のXSLT 勧告と100% 互換性があるわけではありません。)IE 5 を使用して、ブラウザービュー内で問題が発生した場合、IE 6 または以降にアップグレードしてください。使用中のInternet Explorer のバージョンのXSLT に対するサポートを確認してください。

ブラウザービュー機能

ブラウザービューで以下の機能を使用することができます。ブラウザー[972] メニュー、ファイル[742] メニュー、および編集[762] を使用してアクセスすることができます。

- *個別のウィンドウで開く:* ブラウザービューが個別のウィンドウである場合、同じドキュメントの編集ビューで横に並べて表示することができます。これを行うにはメニューコマンド [**ブラウザー | 個別のウィンドウ**]をクリックします。これは2 つのウィンドウ間のブラウザービューを切り替えるトグルです: (i) 個別のウィンドウと(ii) メインウィンドウ内のタブ付けされたビュー。これらのコマンドは(メインウィンドウの下の) **ブラウザービュー** ボタンのドロップダウンメニューで使用することができます。オプションダイアログのビュータブ[1025]によりブラウザービューがデフォルトで個別のウィンドウで表示されるかを設定することができます。
- *「前へ」と「戻る」:*コマンド ブラウザーコマンド は、ブラウザービュー内でロードされたページをナビゲートするために使用されます。これらのコマンドは**ブラウザー** メニュー内にあります。
- *フォントのサイズ:* **ブラウザー** メニューを介して、調節することができます。
- *停止、更新、印刷:* 標準 ブラウザーコマンドは、**ブラウザー** と**ファイル** メニュー内にあります。
- *検索:* テキスト文字列のために検索を有効化します。これらのコマンドは**編集** メニュー内にあります。

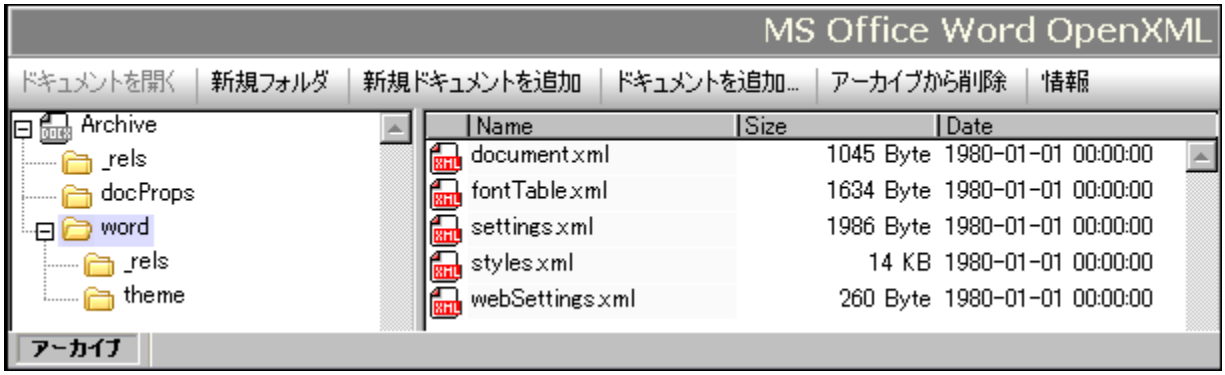
(C) 2015-2021 Altova GmbH

4.7 アーカイブビュー

Archive View ではOffice Open XML (OOXML) ファイルやZIP ファイル、またはEPUB ファイルを開くことができます。OOXML やZIP ファイル、EPUB ファイルの構造を変更するだけでなく、Archive View ではアーカイブ内にある個々のファイルを開き、XMLSpy の編集ビューにより編集を行ない、アーカイブ内に直接保存しなおすことができます。

アーカイブファイルとアーカイブビュー

(OOXML やZIP、EPUB ファイルといった) アーカイブファイルがXMLSpy で作成または開かれたとき562、ファイルは(以下のスクリーンショットにある)Archive View にて開かれます。アーカイブファイルは同時に複数個開くことができ、各アーカイブファイルが個別のArchive View ウィンドウにて表示されます。アーカイブファイルの種類はArchive View の右上に表示されます。以下のスクリーンショットに表示されているアーカイブファイルの種類はMicrosoft Office W ord Open XML です。



フォルダービュー

フォルダービューはArchive View ウィンドウの左側に配置され、圧縮されたアーカイブのフォルダー構造を表示します。各階層においてフォルダーはアルファベット順で表示されます。サブフォルダーを参照するにはフォルダーの左にある+シンボルをクリックしてください。フォルダーの横に+シンボルが無い場合、サブフォルダーは存在しません。フォルダー内にあるドキュメントファイルを閲覧するには、フォルダーを選択することで、メインウィンドウにファイルが表示されます。上のスクリーンショットでは、メインウィンドウにて表示されているドキュメントは`word` フォルダー内部のもので、`_rels` と`theme` の2つのサブフォルダーを含んでいます。

メインウィンドウ

メインウィンドウにはフォルダービューで選択されているフォルダー内に含まれるドキュメントが表示されます。ドキュメントはアルファベット順で表示され、圧縮していない状態でのファイルサイズと更新日時が表示されます。Archive View で表示されているドキュメントをダブルクリックすることで開くことができます。ドキュメントは別のXMLSpy ウィンドウにて開かれます。

コマンドボタン

コマンドボタンはArchive View ウィンドウの上部に位置しています。

- *ドキュメントを開く*：メインウィンドウ内でドキュメントが選択されているときに有効になります。クリックすることで選択されたドキュメントが開かれます。メインウィンドウに表示されているドキュメントをダブルクリックすることでもドキュメントを開くことができます。
- *新規フォルダー*：フォルダービューにて現在選択されているフォルダーに新たなフォルダーを追加します。フォルダービューにてフォルダーが作成されたら、すぐに名前を付けなければなりません。新しいフォルダーは、アーカイブファイルが保存されたときにアーカイブに保存されます。
- *新規ドキュメントを追加*：フォルダービューにて現在選択されているフォルダーに新たなドキュメントを追加します。このボタンをクリックすることでXMLSpy の新規ドキュメントを作成ダイアログが表示されます。新しく作成されたドキュメントは別のXMLSpy ウィ

ンドウにて表示されます。フォルダービューにてファイルが選択されたら、すぐにそのファイルの名前を付けなければなりません。アーカイブファイルが保存されたとき、または編集ウィンドウにてドキュメントが保存された時に作成されたドキュメントは保存されます。

- *ドキュメントを追加...* : 参照ダイアログを起動し、追加するドキュメントを選択することができます。メインウィンドウで現在選択されているフォルダーにドキュメントが追加され、別のXMLSpy ウィンドウにてドキュメントが開かれます。アーカイブ内のドキュメントを保存するには、ドキュメントの編集ウィンドウから保存を行うか、アーカイブファイルを保存する必要があります。
- *アーカイブから削除* : メインウィンドウ、またはフォルダービューにて選択されたドキュメントやフォルダーをアーカイブから削除します。アーカイブファイルが保存されるまで、変更は有効になりません。
- *情報* : 情報ウィンドウの表示を切り替えます。以下を参照ください。

情報ウィンドウ

情報ボタンをクリックすることで、情報ウィンドウの表示を切り替えることができます。情報ウィンドウでは、アーカイブに含まれるファイルの数や、圧縮時と非圧縮時のファイルサイズ、また圧縮率といったアーカイブファイルに関する一般的な情報が表示されます。

4.8 共通のショートカット

一般的に使用される編集コマンドのデフォルトのショートカットは以下にリストされています。カスタマイズダイアログのキーボードタブ[1003]内でデフォルトのショートカットを変更することができます。

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

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

ファイルとアプリケーションコマンド

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

その他のキー

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

編集コマンド

CTRL + A	すべて選択
CTRL + F	検索

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

(C) 2015-2021 Altova GmbH

5 XML

このセクションではXMLSpy にてXMLドキュメントを操作する方法が記述されています。XMLに関する以下のような側面がカバーされます:

- XMLドキュメントを作成、開く、保存する方法 230 。このセクションではファイルの作成に関するXMLSpy の重要な設定についても説明されます。
- XMLドキュメントはテキストビュー 234 、グリッドビュー 236 、またはAuthentic ビュー 268 にて編集することができます。使いやすいビューを選択して、編集時にビューの切り替えを行うことができます。各ビューではそれぞれのビューが持つ利点を活かすことができます。
- XML フラグメント 273 をXMLドキュメントに外部フォルダーから簡単に素早く追加することができます。
- XMLドキュメントの入力ヘルパー 270 が持つ独自の機能に関する解説。
- XSLT やXQuery を含むXMLドキュメントの操作方法 275 。これらの処理を行うのに関連したXMLSpy が持つ様々な機能が説明されます。PDF フォント 277 に関するセクションでは、PDF 出力を生成する際のフォントの処理方法について説明されます。
- XMLドキュメントで作業をする際に使用できるその他の機能 279 を紹介します。

Altova Web サイト: XML エディター

5.1 XMLドキュメントを作成、開く、保存する

XMLドキュメントを作成、開く、そして保存するときには、以下のことがらが関連します:

- テキストビュー、グリッドビュー、Authentic ビューのうち、どのビューでXMLドキュメントは開かれているのか
- 新規XMLドキュメントが作成されたとき、スキーマ(XMLスキーマまたはDTD)が自動的に、もしくは手動で割り当てられるか、または割り当てられないのか
- XMLドキュメントにスキーマが割り当てられている場合、ドキュメントを開き保存するときに検証は自動的に行われるのか

デフォルトビュー

アプリケーション全体で有効になる設定により、(新規または既存に関わらず)XMLファイルが開かれるビューを指定することができます。これらの設定はオプションダイアログ「**ツール | オプション**」に収められています。

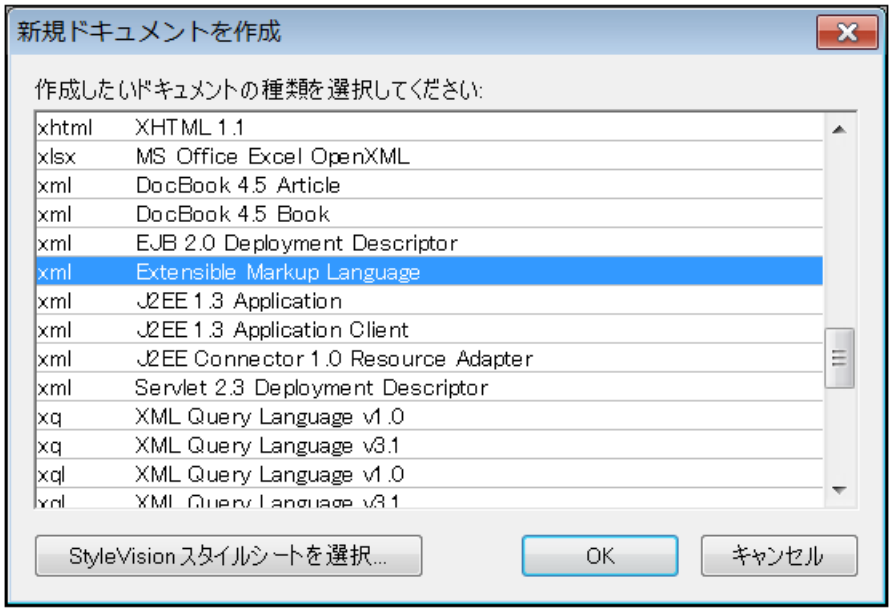
オプションダイアログの*ファイルの型*セクションにて `.xml` を選択し、デフォルトビューから目的の編集ビュー(テキストビューまたはグリッドビュー)を選択します。スキーマビューはXMLスキーマでしか選択できないことと、ブラウザービューは表示のためのビューであり、編集ビューでは無いことに注意してください。

ファイルの型 タブでは、選択されたファイルの種類に対してXMLSpy をデフォルトのエディターに設定することもできます。

StyleVision Power Stylesheet(SPS) ファイルが割り当てられている場合、Authentic ビューでもXMLドキュメントを編集することができます。割り当てられたSPS ファイルと一緒にXMLファイルが開かれた場合、Authentic ビューで直接開くように設定することができます。オプションダイアログのビューセクションにある、*常にファイルを Authentic ビューで開く*にチェックを入れてください。このオプションにチェックが入っていない場合、*ファイルの型* タブ `.xml` ファイルで指定されたデフォルトビューにてファイルが開かれます。

スキーマの割り当て

新規XMLファイルを作成するには、「**ファイル | 新規作成**」メニューコマンドを選択してください。新規ドキュメントを作成ダイアログ(以下のスクリーンショット)が表示されます。



XMLドキュメントとして選択できるアイテムが複数あることに注意してください。*Extensible Markup 言語* として記述されているオプションにより一般的なXMLドキュメントが作成されます。その他のオプションでは、例えばDocBook といったスキーマと、XMLドキュメントが既に関連付けられていたりします。関連付けされているこれらのオプションを選択すると、(i) 対応するスキーマが自動的に割り当てられ、(ii) 割り当てられたスキーマにより妥当なドキュメントのスケルトンがデフォルトでXMLドキュメントに作成されます。XMLドキュメントのスケルトンは独自に作成することができます。アプリケーションフォルダーの `Template` フォルダーに保存すれば、スケルトンドキュメントが新規ドキュメントを作成ダイアログにて使用できるようになります。

一般的なExtensive Markup 言語 をドキュメントの種類として選択した場合、ドキュメントにスキーマ(DTD またはXML スキーマ)を割り当てるかどうか質問されます。ここでは、スキーマを参照するか、スキーマを割り当てること無くXMLドキュメントの作成にとりかかることができます。

勿論、「**DTD / スキーマ**」メニューを使えば、ドキュメントの編集中いつでもスキーマを割り当てることができます。

自動検証

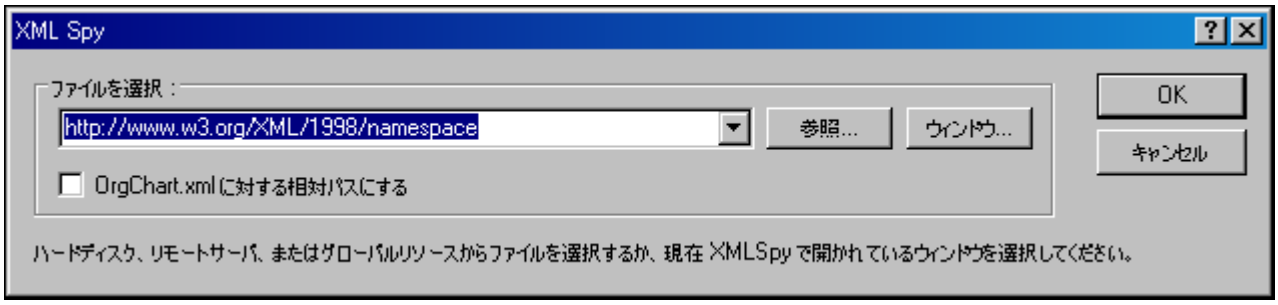
既存のXMLドキュメントにスキーマが割り当てられている場合、ドキュメントを開いたり保存する際に自動的に検証を行うことができます。オプションダイアログ(「**ツール | オプション**」)の*ファイル*セクションから設定を行うことができます。

*ファイル*タブ内の自動検証設定をファイルの型のセクションの設定と組み合わせることで、特定のファイルの型を自動的に検証することができます。これら2つのタブにある設定を行うことで、

5.2 スキーマの割り当てと検証

Altova ウェブサイト： XML バリデーター、XML 検証

XML ドキュメントの作成時 230 にスキーマ(DTD または XML スキーマ) の割り当てを行うことができます。スキーマの割り当てや変更は、「DTD / スキーマ」メニュー以下にある「DTD の割り当て」または「スキーマの割り当て」コマンドにより、いつでも行うことができます。



ダイアログ内の相対パスにチェックを入れることで、XML ドキュメントへ挿入されるスキーマファイルへのパスを XML ファイルから見た相対パスにすることができます。

スキーマのグローバルリソース

グローバルリソースとはファイルやフォルダーへのエイリアスです。ターゲットファイルまたはフォルダーは、(「ツール | アクティブな構成」により) グローバルリソースのアクティブ構成を変更することでダイナミカルに変更することができます。グローバルリソースを使えば割り当てられた複数のスキーマの切り替えを瞬時に行うことができ、試験を行うときなどに力を発揮します。グローバルリソースの使用方法については、Altova グローバルリソース 660 を参照ください。

XML スキーマと DTD

XML スキーマには無い DTD の便利な機能の1つにエンティティがあります。もし検証されたスキーマにてエンティティを使用したい場合は、DOCTYPE 宣言を XML ドキュメント内で行ない、その中でエンティティの宣言を行う必要があります。

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE OrgChart [
  <!ENTITY name-int "value">
  <!ENTITY name-ext SYSTEM "extfile.xml">
]>
<OrgChart xmlns="http://www.xs.com/org"
          xsi:schemaLocation="http://www.xs.com/org OrgChart.xsd">
  ...
</OrgChart>
```

DTD 内部でエンティティの宣言を行った後に、それらを XML ドキュメント内で使用することができます。ドキュメントは整形式ならびに妥当なものとなります。しかし外部で解析されたエンティティは Authentic ビューでサポートされません。

スキーマの定義へ移動

XML ドキュメントが開かれていれば、「DTD/スキーマ」メニュー以下の DTD へ移動または スキーマへ移動コマンドをクリックすることで、ベースとなっている DTD または XML スキーマを直接開くことができます。更に、XML ドキュメント内のノードにカーソルを移動させ、「DTD/スキーマ | 定義へ移動」コマンドを使うことで、ノードのスキーマ定義に直接移動することができます。

検証と整形式のチェック

ドキュメントの検証や整形式のチェックを行うには、検証 (F8) や**整形式のチェック** (F7) コマンドを XML メニューまたはツールバーから使用します。エラーや警告がある場合は、メッセージウィンドウに表示されます。XML ドキュメントが不正の場合、スキーマ内にある情報をベースに、スマートフィックスによるエラーの修正 841 を行うことができます。

RaptorXML Server 683 を使用して XML ドキュメントを検証することができます。

5.3 テキストビュー内の XML

XMLSpy は編集ビューセクション内のテキストビュー 122 内で説明されているテキストビュー内の一般的に使用可能な編集機能に加え、特殊な XML テキスト編集機能を提供します。

- テキストのコメントインとアウト 234
- XML 文字のエスケープとエスケープの解除 234
- ファイルパスの挿入 235
- XInclude を介したXML フラグメントの挿入 235
- XPath とXpointer 式をクリップボードにコピーする 235

テキストのコメント イン/アウト

XML コメントの開始とコメントの終了区切り文字 <!-- と--> をそれぞれ使用してXML ドキュメント内のテキストをコメント アウト することができます。XMLSpy 内でこれらの区切り文字は **編集 | コメント イン/アウト** メニューコマンドを使用して簡単に挿入することができます。

テキストのブロックをコメント アウト するには、コメント アウト するテキストを選択し **コメント イン/アウト** を **編集** メニューまたは選択されたテキストを右クリックすることにより表示されるコンテキスト メニューから選択します。コメント アウト されたテキストはグレーで表示されます（*以下のスクリーンショットを参照*）。

```
<Department>
    <Name>Administration</Name>
    <Person>
    <Person>
    <Person>
    <!--<Person>
        <First
        <Last></Last>
        <PhoneExt></PhoneExt>
        <EMail></EMail>
        <LeaveTotal></LeaveTotal>
        <LeaveUsed></LeaveUsed>
        <LeaveLeft></LeaveLeft>
    </Person>-->
</Department>
```

テキストのコメントされたブロックをアンコメントするには、コメント区切り文字を **除外** してコメントされたブロックを選択し、コマンド **コメント イン/アウト** を **編集** メニューまたは選択されたテキストを右クリックすると表示されるコンテキスト メニューから選択します。コメント区切り文字はテキストがグレーアウトされなくなると削除されます。

空のラインに関するメモ

XML ドキュメント内では、ビューを変更またはドキュメントを保存する際に空のラインは破棄されます。空のラインを保持するには、これをコメントの区切り文字で囲みます。

XML 文字のエスケープとエスケープの解除

（*下にリストされる*）5 個の XML 特殊文字 対応する（*下にリストされる*）エンティティ リファレンスに従いテキストのブロックをハイライトし、コンテキスト メニュー コマンド **XML 文字をエスケープ** または **XML 文字をアンエスケープ** を選択してエスケープおよびアンエスケープすることができます。テキストのそのブロック内の XML 特殊文字は選択されたコマンドに従いエスケープまたはアンエスケープされます。

<	<
>	>
&	&

(C) 2015-2021 Altova GmbH

```
'      &apos;
"      &quot;
```

例:

<a> という文字列に対して「XML 文字のエスケープ」コマンドを適用すると <a> という文字列への変換が行われ、
<a> という文字列に対して「XML 文字のエスケープの解除」コマンドを適用することで、文字列が <a> に変換されます。

ファイル パスの挿入

編集 | ファイル パスを挿入 768 コマンドにより、目的のファイルを参照し、そのファイル パスを XML ドキュメント内で選択されている場所に挿入することができます。このコマンドによりファイル パスを素早く正確に入力することができます。詳細については コマンドの記述 768 を参照ください。

XInclude を介したXML フラグメントの挿入

編集 | XInclude を挿入 769 により、XInclude により XML ドキュメント全体のコンテンツまたはフラグメントを編集中の XML ドキュメントに挿入することができます。このコマンドにより、XML ドキュメント全体、または XML ドキュメントのフラグメントを(XInclude メカニズムの XPointer 拡張を介して)素早く正確に入力することができます。詳細については、コマンドの記述を参照ください。詳細については コマンドの記述 769 を参照ください。

XPath ならびに XPointer 表現をクリップボードにコピー

編集 | XPath をコピー 768 および 編集 | XPointer をコピー 768 を使うことで、選択されているノードの XPath と XPointer 表現をそれぞれクリップボードにコピーすることができます(ノードの識別は XML ドキュメント内の位置により行われます)。これにより目的とするノードの正確な XPath ならびに XPointer 表現を取得することができます。

例えば、Offices というドキュメント要素以下にある3番目の Office 要素がテキストビューまたはグリッドビューにて選択されている場合を考えてみましょう。この場合 /Offices/Ofice[3] という XPath 条件式がコピーされます。コピーされた XPointer 式は、3番目の Office 要素の前に Office 要素が他の名前を持つ兄弟を持たない場合 element(/1/3) になります。

コピーされた条件式をは任意の場所に挿入することができます。例えば、XPath 条件式を XSLT スタイルシートへ挿入し、XPointer 条件式を xinclude 要素の href 属性内に挿入することができます。

コマンドのより詳細な情報については、ユーザーリファレンス内の対応するセクションを参照ください。

5.4 グリッドビュー内の XML

グリッドビュー[141] はXML ドキュメント の階層構造を展開または折りたたみ可能なネストされたコンテナーのセットを介して表示します。これはドキュメントの構造の明確なピクチャを提供します。グリッドビューでは、ドキュメント構造を簡単に変更でき、コンテンツを簡単に編集することができます。

XML	<?xml version="1.0" encoding="UTF-8"?>				
Company					
	xmlns	http://my-company.com/namespace			
	xmlns:xsi	http://www.w3.org/2001/XMLSchema-instance			
	xsi:schemaLocation	http://my-company.com/namespace AddressLast.xsd			
	Address				
		xsi:type	US-Address		
		Name	US dependency		
		Street	Noble Ave.		
		City	Dallas		
		Zip	04812		
		State	Texas		
	Person (3)		Manager	Degree	Programmer	First
		1	false	MA	true	Alfred
		2	true	Ph.D	false	Colin
		3	true	BA	false	Fred

上のスクリーンショット内で、ドキュメントがグリッドフォームで表示されていることに注意してください。ノードがコンテンツを含むことが可能な場合、ノードは(*テキスト* ノードの場合と同様)直接コンテンツを含む、または(*要素* ノードの場合と同様)以下の2つのフィールドに分割されます: ノード名とノードコンテンツ。ノード名は太字で表示され、ノードコンテンツは通常の書体で表示されます。

グリッドビューツールバー

アプリケーションウィンドウの上のツールバーエリアにあるグリッドビューツールバーを使用してグリッドビュー内の編集に関連するコマンドにアクセスすることができます(*下のスクリーンショット*)。名前とショートカットを確認するにはツールバーアイコンにマウスをポイントしてください。

メモ これらのコマンドはXML メニュー[812] 内で使用可能です。

グリッドビュー機能の概要:

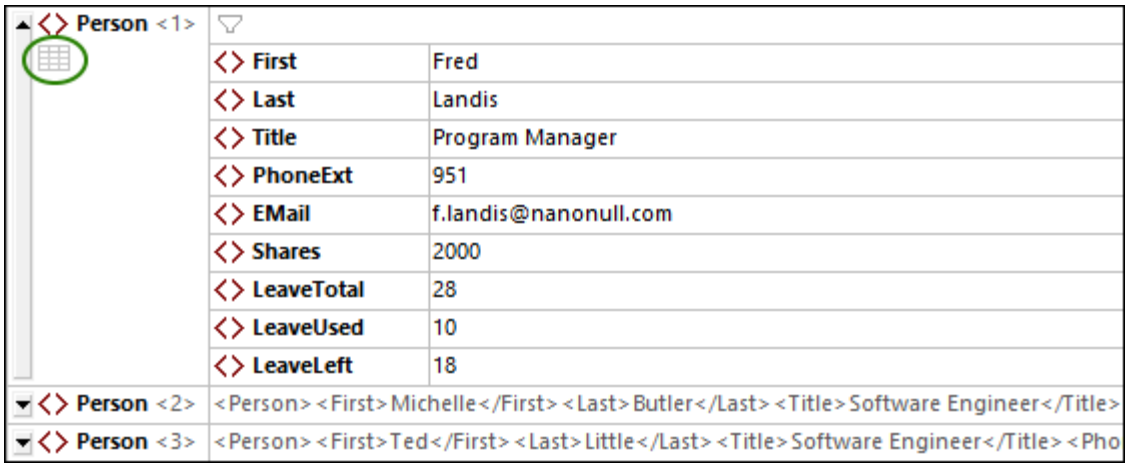
グリッドビュー機能の概要に関しては以下を参照してください:

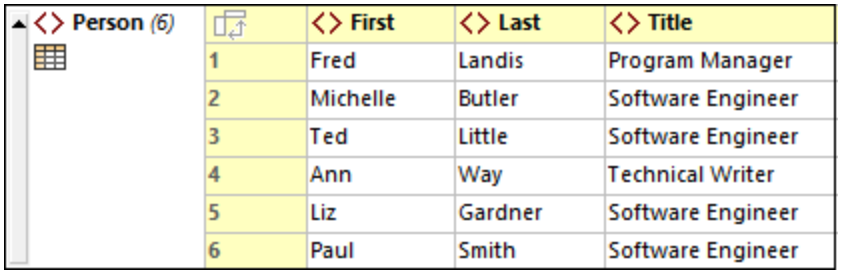
- グリッドビューの表示機能を表示するドキュメントの表示[238]。
- ドキュメントの構造がグリッドビューでどのように表示されるかを記述し、ドキュメント構造を変更するためにグリッドビュー機能を使用する方法を説明するドキュメントの構造[241]。
- グリッドビュー内でコンテンツの編集と検証を表示するドキュメントコンテンツ[243]
- グリッドビューに適用する設定をリストし説明するグリッドビュー設定[265]。

テーブルとして表示する

繰り返し要素は*標準*のグリッドビューでドキュメント順に垂直下法で順番に表示されています(*左下のスクリーンショット参照*)。しかしながら、繰り返し要素を *テーブルの行* として表示すると追加編集機能が与えられます。下のスクリーンショットでは `Person` 要素は繰り返し要素です。左のスクリーンショットは標準のグリッドビューを表示しています。最初の `Person` 要素が展開されて表示され、続くインスタンスは折りたたまれて表示されます。右のスクリーンショットは繰り返し `Person` 要素をテーブルの行として表示しています。

テーブルビューに切り替えるには(左下のスクリーンショット内で緑色の丸で囲まれている) **テーブルモード** *アイコン をクリックします*。テーブルディスプレイに切り替えると、アイコンは色で表示されます (*右下のスクリーンショット*)。





テーブルビューはその行と列全体がテーブル内の他の列と行に相対的に操作可能になるユニークな編集の利点を提供します。これにより列の値上のテーブル行の並べ替えなどの操作が可能になります。例えば、右上のスクリーンショットでは、6個の `Person` 要素がグリッドビューツールバー[236]内の単一のコマンドにより `Last` 子要素をベースに並べ替えられることができます。これはXML ノードセットを並べ替えるための通常の方法であるXSLT 変換を実行するよりもシンプルです。

詳細に関しては、テーブル表示[250]のトピックを参照してください。

フィルターとフォーミュラ

グリッドビューは2つのユニークで役に立つ機能を提供します。

- グリッドビューで表示されているデータのフィルターを可能にするフィルター[254]。ドキュメント内で1つまたは複数の要素にフィルターを設定することができます。
- フォーミュラ[255] により結果を計算、またはドキュメント内に保管可能なノードセットを生成することができます。

詳細に関しては、関連するトピックを参照してください。

グリッドビュー内でのイメージ

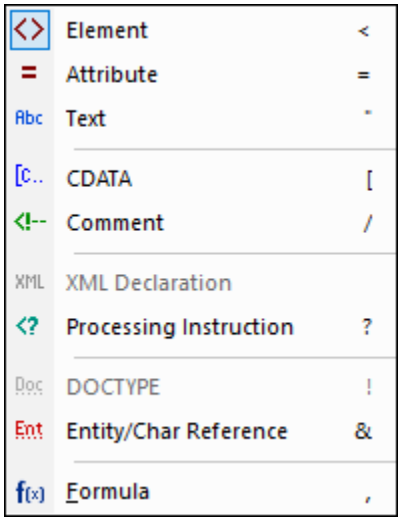
イメージ はグリッドビュー内でグラフィックとして表示することができます。この方法についての詳細はグリッドビュー内のイメージの表示 258 を参照してください。

5.4.1 ドキュメントの表示

グリッドビュー内ではXMLドキュメントは階層的に表示されます（*下のスクリーンショットを参照*）。

XML	<?xml version="1.0" encoding="UTF-8"?>	
OrgChart		
	xmlns	http://www.xmlspy.com/schemas/orgchart
	xmlns:ipo	http://www.altova.com/IPO
	xmlns:ts	http://www.xmlspy.com/schemas/textstate
	xmlns:xsi	http://www.w3.org/2001/XMLSchema-instance
	xsi:schemaLocation	http://www.xmlspy.com/schemas/orgchart NanonullOrg.xsd
	CompanyLogo	
		href: nanonull.gif
	Name	Organization Chart
	Office <1>	
		Name: Nanonull, Inc.
		Desc
		para <1>: <para>The company was established
		para <2>
		Abc Due to the fact that nanoelectroni publicity in the company's early ye
		Location: US
		Address
		ipo:street: 119 Oakstreet, Suite 4876
		ipo:city: Vereno
		ipo:state: DC
		ipo:zip: 29213
		Phone: +1 (321) 555 5155 0
		Fax: +1 (321) 555 5155 4
		EMail: office@nanonull.com

各グリッドラインは要素、属性、コメントまたはテキストなどの1つの（ノードとして既知の）XML構造アイテムを表示します。XMLドキュメント内で使用可能な異なるノード型はアイコンと共に下のスクリーンショットで表示されています。グリッドビューでは、ノードの型はグリッドセル内の左側にあるアイコンで示されています。型 *要素*、*属性* と *処理命令*のノードはキーと値を持ちますが、*テキスト*、*CDATA* および *コメント* は値のみを持っています。例えば、要素ノードは名前（自身のキー）とコンテンツ（自身の値）、を持ちますが、テキストノードは値のみを持ちます。



グリッドビュー表示の機能

以下の機能に注意してください：

- Ctrl + **マウスのホイール** または Ctrl+[**プラス**] と Ctrl+[**マイナス**]を使用してグリッドをズームすることができます。
- 同じ名前を持つ兄弟要素に 1 から始まる番号が与えられます。上のスクリーンショットでは、例えば、1 の番号を持つ `Office` 要素が `Office` 要素のシーケンスの最初になります。同じ名前の兄弟要素を持たないためスクリーンショットの他の要素は番号を付けられていません。
- ノードシンボルの左側の矢印アイコンを押してノードを展開または折りたたむことができます *(上のスクリーンショットを参照)*。ノードが折りたたまれると、コンテンツは単一のライン内のテキストで表示されます。
- 複数のコンポーネントを同じレベルで選択すると、これらを **Shift** を押し、選択されたコンポーネントの矢印をクリックして展開/折りたたむことができます。
- ワードラップが グリッドビュールールバー 236 を介してオンに切り替えられている場合、セルの幅より先長いテキストを含むすべてのセルはラップされます。ワードラップを グリッドビューツールバー内の 236 アイコンをクリックしてオフに切り替えます。
- フォーミュラ 492 はXMLSpy に固有です。

ツールバー/セル内のコマンドボタン

XML グリッドビュー内では、グリッドセル内または グリッドビューツールバー 236 内のボタンは現在の選択範囲（セルまたは列）のコンテキストで使用可能な機能を提供します。これらのコマンドの使用は一般的に使用される操作を更に素早く簡単に実行する方法です。使用可能なコマンドは下のテーブルにリストされています。表示に関連したコマンドは青い背景と共に表示されています。ドキュメント（構造とコンテンツ）の編集のためのコマンドはベージュの背景で表示されています。特定のセルのコンテキスト内の追加コマンドのためにセルの コンテキストメニュー 500 を使用してください。メモ セルボタンまたはツールバーアイコンをポイントするとコマンドの名前がツールヒントとして表示されます。

	セルコンテンツに従いグリッド列の幅を最適化
/	色で表示されている場合 テーブルの表示 488 がオンに切り替えられています。それ以外の場合はオフです。表示するためにクリックします。
/	テーブルの左側のセル内のコマンドをトグルします。行を列に、または列を行に切り替えます。
/	色で表示されている場合、テーブル、オブジェクト、または配列のためのフィルター 492 が有効化されています。それ以外の場合フィルターは無効化されています。無効化/有効化するためにクリックします。式を編集するには、ダブルクリックします。

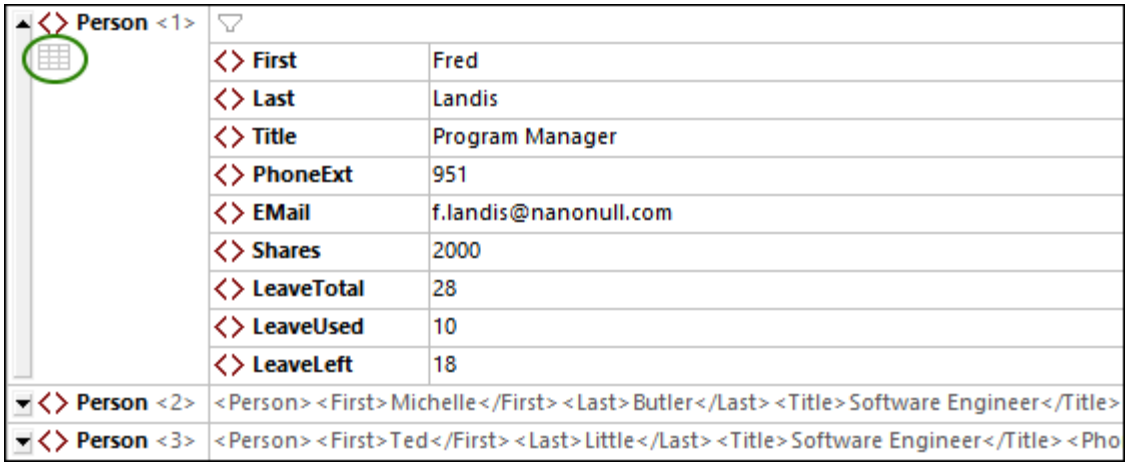
WR AP	セルをワードラップするためのトグルコマンド。選択されると、ワードラップがオンになります。デフォルトでは、アイテムのコンテンツのみがラップされます。アイテムの名前を追加でラップする場合、このオプションをアイコンのドロップダウンリストから選択します。
100%	グリッドビューのズームレベル
	テーブルビュー内で列ヘッダーが選択されると有効化されます。ボタンはテーブルの行を列のコンテンツの降順または昇順の順序にそれぞれ並べ替えます。

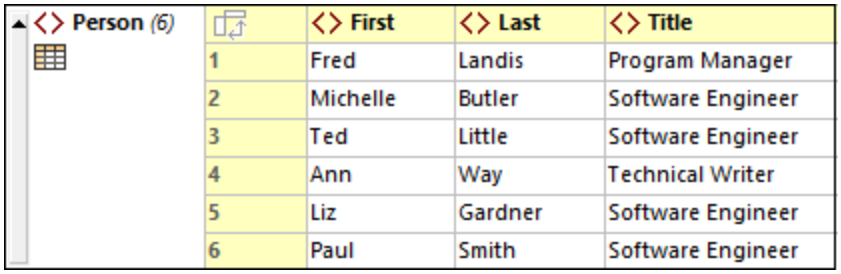
メモ 他のツールバーコマンドの詳細に関しては、セクショングリッドビュー内のコンテキストメニュー 262を参照してください。

テーブルとして表示する

繰り返し要素は*標準*のグリッドビューでドキュメント順に垂直下法で順番に表示されています(*左下のスクリーンショット参照*)。しかしながら、繰り返し要素を *テーブルの行* として表示すると追加編集機能が与えられます。下のスクリーンショットでは`Person` 要素は繰り返し要素です。左のスクリーンショットは標準のグリッドビューを表示しています。最初の`Person` 要素が展開されて表示され、続くインスタンスは折りたたまれて表示されます。右のスクリーンショットは繰り返し`Person` 要素をテーブルの行として表示しています。

テーブルビューに切り替えるには(左下のスクリーンショット内で緑色の丸で囲まれている) **テーブルモード** *アイコン をクリックします*。テーブルディスプレイに切り替えると、アイコンは色で表示されます (*右下のスクリーンショット*)。





テーブルビューはその行と列全体がテーブル内の他の列と行に相対的に操作可能になるユニークな編集の利点を提供します。これにより列の値上のテーブル行の並べ替えなどの操作が可能になります。例えば、右上のスクリーンショットでは、6個の`Person` 要素がグリッドビューツールバー 236内の単一のコマンドにより`Last` 子要素をベースに並べ替えられることができます。これはXML ノードセットを並べ替えるための通常の方法であるXSLT 変換を実行するよりもシンプルです。

詳細に関しては、テーブル表示 250のトピックを参照してください。

5.4.2 ドキュメント構造

グリッドビューではXMLドキュメント構造を視覚的に編集することができます。例えば、ドキュメント構造の個別のセグメントを折りたたむまたは展開する、ノードを挿入、追加および削除する、異なる個所にノードをドラッグアンドドロップする、および1つの型を他の型に変換することができます。

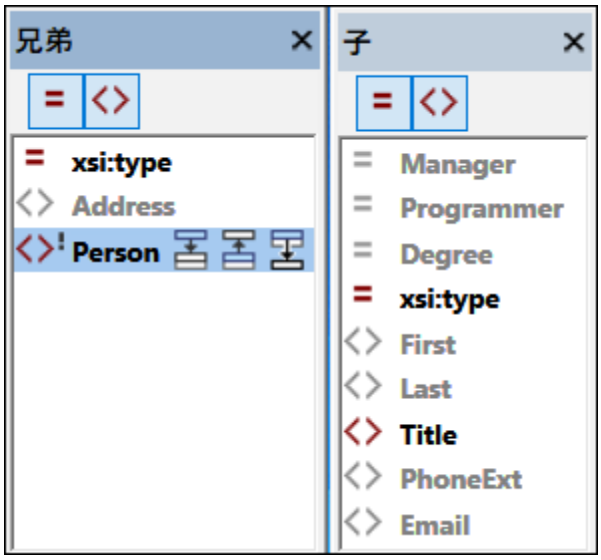
新規のノードの追加

ドキュメントに新規のノードを追加する方法が2通りあります:

- 兄弟と子入力ヘルパーは選択された箇所に特定のノードを入力します。これが作動するにはXMLドキュメントがスキーマ(DTDまたはXMLスキーマ)に関連付けられている必要があります。例えば、メインウィドウ内でノードを選択する場合、(スキーマに従い) その箇所で許可されている兄弟と子がそれぞれ兄弟と子入力ヘルパーに表示されます。入力ヘルパーを使用して追加するノードを選択することができます。
- 兄弟、子、または親として定義されていない新規のノードを追加することができます。グリッドビューセルのコンテキストメニュー内でこれを行うコマンドを使用することができます。またXML メニュー 812 とグリッドビューツールバー 236 でも使用することができます。

兄弟と子入力ヘルパー

XMLドキュメントがスキーマ(DTD またはXML スキーマ)に関連づけられている場合、使用可能な選択されたノードの兄弟と子が兄弟と子入力ヘルパーに表示されます。これらの入力ヘルパー内で要素と属性のトグルの可視性を対応するツールバーアイコンをクリックすることによりオンとオフに切り替えることができます。必須のノードは感嘆符と共に表示されています。灰色表示されているノードは追加することができません。追加できない理由は以下の理由の一つです: (i) ノードが既に追加されており、下のスクリーンショット内の`Address` 兄弟の場合で示されている通りスキーマにより更なるノードが許可されている場合。(ii)下のスクリーンショット内の`First` 子が追加された後に追加可能になる`Last` 子の場合で示されている通り灰色表示されているノードの前に他のノードが追加される必要がある場合。



ノードを以下のように追加します:

- *兄弟:* 入力ヘルパー内で、兄弟として追加するノードを選択します。グリッド内で選択されているノードの後または前に兄弟を追加するか、または選択されたノードの兄弟の最後として付加するかにより適切なアイコンします *(上の左側のスクリーンショットを参照)*。
- *子:* 子として追加するノードをダブルクリックします。

新規のノードの追加

ドキュメント（すなわちグリッドセル）内のノードが選択されるとからのノードを兄弟、子、親として追加することができます。これらの操作のためのコマンドは下にリストされており、以下で使用することができます: (I) セルのコンテキストメニュー(ii) XML メニュー[812]。および(iii) グリッドビューツールバー[236]。

コマンド	ショートカット
後に（兄弟）を挿入	Ctr+Enter
前に（兄弟）を挿入	Ctr+Shift+Enter
付加（兄弟）	Ctr+Alt+A
子の追加	Ctr+Alt+Enter
要素内のラップ	Ctr+Alt+W

以下の点に注意してください：

- 新規のノードはデフォルトで空の要素として作成されます。ノード型を後に変更[243] することができます。
- **要素内のラップ** コマンドは現在のノードの周りに要素ノードを作成します。この要素は現在のノードの新規の親となります。

標準のWindows メカニズムを使用して構造を変更します。

グリッドビューではWindows メカニズムを使用してドキュメント構造を変更することもできます:

- *削除:* コンポーネントを選択して **Delete** キーを使用して削除します。
- *移動:* コンポーネントを選択してドラッグアンドドロップを使用して新規のロケーションにドロップします。

テーブルとして表示する

繰り返し要素は*標準*のグリッドビューでドキュメント順に垂直下法で順番に表示されています（*左下のスクリーンショット参照*）。しかしながら、繰り返し要素を *テーブルの行* として表示すると追加編集機能が与えられます。下のスクリーンショットでは `Person` 要素は繰り返し要素です。左のスクリーンショットは標準のグリッドビューを表示しています。最初の `Person` 要素が展開されて表示され、続くインスタンスは折りたたまれて表示されます。右のスクリーンショットは繰り返し `Person` 要素をテーブルの行として表示しています。

テーブルビューに切り替えるには（左下のスクリーンショット内で緑色の丸で囲まれている）**テーブルモード** *アイコン をクリックします。テーブルディスプレイに切り替えると、アイコンは色で表示されます（右下のスクリーンショット）。*

Person <1>		
	First	Fred
	Last	Landis
	Title	Program Manager
	PhoneExt	951
	EMail	f.landis@nanonull.com
	Shares	2000
	LeaveTotal	28
	LeaveUsed	10
	LeaveLeft	18
Person <2>	<Person><First>Michelle</First><Last>Butler</Last><Title>Software Engineer</Title>	
Person <3>	<Person><First>Ted</First><Last>Little</Last><Title>Software Engineer</Title><Pho	

Person (6)		First	Last	Title
	1	Fred	Landis	Program Manager
	2	Michelle	Butler	Software Engineer
	3	Ted	Little	Software Engineer
	4	Ann	Way	Technical Writer
	5	Liz	Gardner	Software Engineer
	6	Paul	Smith	Software Engineer

テーブルビューはその行と列全体がテーブル内の他の列と行に相対的に操作可能になるユニークな編集の利点を提供します。これにより列の値上のテーブル行の並べ替えなどの操作が可能になります。例えば、右上のスクリーンショットでは、6個の`Person`要素がグリッドビューツールバー236内の単一のコマンドにより`Last`子要素をベースに並べ替えられることができます。これはXMLノードセットを並べ替えるための通常の方法であるXSLT変換を実行するよりもシンプルです。

詳細に関しては、テーブル表示250のトピックを参照してください。

5.4.3 ドキュメントコンテキスト

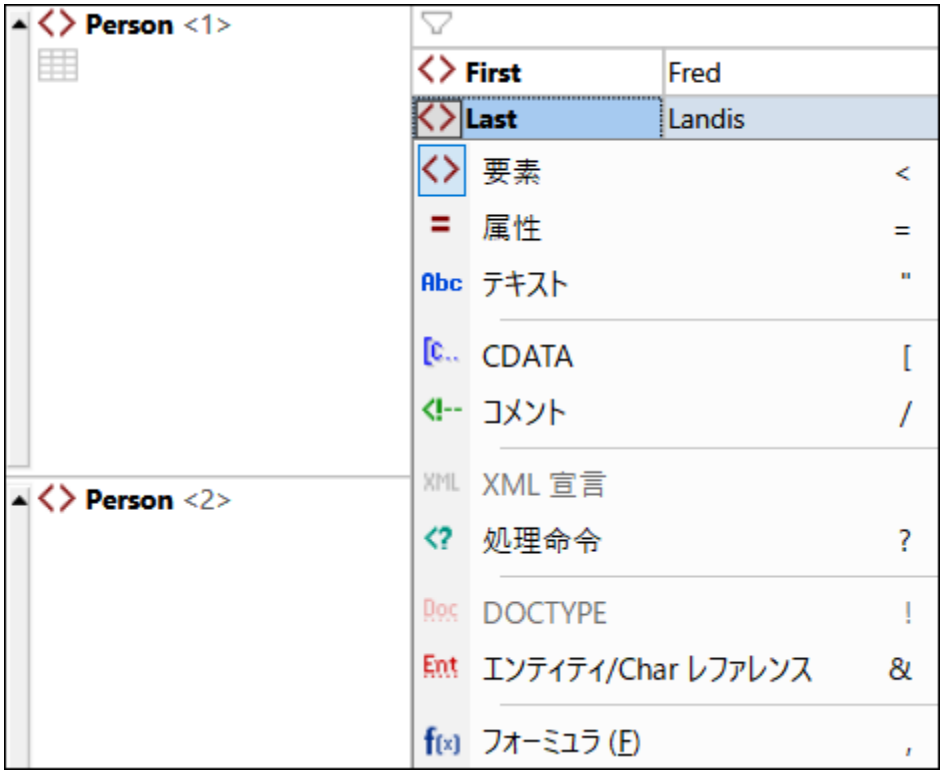
グリッドビュー内のXMLコンテンツの編集はとても理解しやすいです: コンテンツフィールドの内部をダブルクリックし、必要に応じてコンテンツを編集します。ノードの型アイコンをクリックし表示されるメニューから他の型を選択することによりノード型は素早く変更されます。エンティティ入力ヘルパーを介してエンティティを挿入することができます。グリッドビューは更に検証と検索と置き換え機能もオファーしています。

ノードの型の選択と変換

グリッドビュー内のセルはXMLドキュメントのノードを含んでいます。よりよく理解するために型は以下のように分類されています:

- *名前-値型:* 要素、属性、処理命令(PI)。この型のノードは名前と値を有しています。
- *値型:* テキスト、CDATA、コメント、エンティティ/文字レファレンス。この型のノードは値のみを有します。
- *定義型:* XML宣言、DOCTYPE(内部または外部DTD)。これらのノードはXMLドキュメントのプロパティを定義します。
- *XMLSpyに固有の特別なコンテナー型:* フォーミュラ。

値の前のシンボルにより示される値の型(*下のスクリーンショットを参照*)。型を変更するには、シンボルをクリックして、表示されるメニューから型を選択します。または、セルを右クリックして、表示されるコンテキストメニューの型 サブメニューから型を選択します。型のシンボルとショートカットは下のスクリーンショットに表示されています:



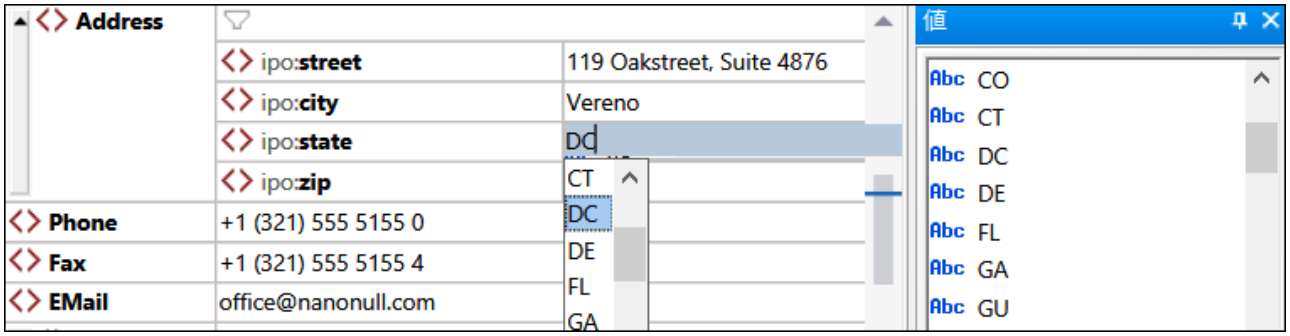
以下の点に注意してください：

- 型 *要素*、*属性* および *PI* は名前と値を持ちますが型 *テキスト*、*CDATA* および *コメント* は値だけを持ちます。例えば、要素ノードは名前とコンテンツを持ちますが、テキストノードは値のみを持ちます。
- 型の変換は元のキーと値を保持しようと試みます。例えば、要素を属性に変換すると属性は (i) 要素と同じ属性 (ii) 要素のコンテンツと可能な限り同じの値を持ちます。
- 要素の子ノードのシーケンス内では属性は常に最初にリストされます。この結果、型変換はノードの並べ替えを引き起こします。
- (テキストと太字および斜体要素を含む パラグラフ要素など要素の子が混在する文字データ) 複合型コンテンツの場合 連続したテキストノードは許可されておらず自動的に連結される可能性があります。
- 複数のノードが単一型に変換されるために選択される場合、この変更が先に行われるか否か、または、警告を与えるかを決定するオプションを設定することができます。

自動補完

編集される XML ドキュメントがスキーマに関連している場合自動補完が有効化されます。

自動補完はカーソルの位置で入力オプションを提供します。これは以下を使用して行うことができます (*下のスクリーンショット参照*)。これらのオプションはスキーマ内の定義をベースにしており (i) メインウィンドウ内のポップアップ (ii) 入力ヘルパーを介して提供されます。ポップアップと入力ヘルパーはそれぞれカーソルのロケーションで有効なエントリのリストを表示します。ポップアップリスト内のエントリを移動するには矢印キーを使用してください。エントリをポップアップウィンドウから選択、または入力ヘルパー内のエントリをダブルクリックして挿入します。



ツールバー/セル内のコマンドボタン

XML グリッドビュー内では、グリッドセル内または[グリッドビューツールバー](236) 内のボタンは現在の選択範囲（セルまたは列）のコンテキストで使用可能な機能を提供します。これらのコマンドの使用は一般的に使用される操作を更に素早く簡単に実行する方法です。使用可能なコマンドは下のテーブルにリストされています。表示に関連したコマンドは青い背景と共に表示されています。ドキュメント（構造とコンテンツ）の編集のためのコマンドはベージュの背景で表示されています。特定のセルのコンテキスト内の追加コマンドのためにセルの[コンテキストメニュー](500)を使用してください。メモ　セルボタンまたはツールバーアイコンをポイントするとコマンドの名前がツールヒントとして表示されます。

	セルコンテンツに従いグリッド列の幅を最適化
/	色で表示されている場合 [テーブルの表示](488) がオンに切り替えられています。それ以外の場合はオフです。表示するためにクリックします。
/	テーブルの左側のセル内のコマンドをトグルします。行を列に、または列を行に切り替えます。
/	色で表示されている場合、テーブル、オブジェクト、または配列のための[フィルター](492) が有効化されています。それ以外の場合フィルターは無効化されています。無効化/有効化するためにクリックします。式を編集するには、ダブルクリックします。
	セルをワードラップするためのトグルコマンド。選択されると、ワードラップがオンになります。デフォルトでは、アイテムのコンテンツのみがラップされます。アイテムの名前を追加でラップする場合、このオプションをアイコンのドロップダウンリストから選択します。
100%	グリッドビューのズームレベル
	テーブルビュー内で列ヘッダーが選択されると有効化されます。ボタンはテーブルの行を列のコンテンツの降順または昇順の順序にそれぞれ並べ替えます。

メモ　他のツールバーコマンドの詳細に関しては、セクション[グリッドビュー内のコンテキストメニュー](262)を参照してください。

変更中に検証

編集中に検証 モードはデフォルトでオフに切り替えられています。オンに切り替えられていると、グリッドビュー内でのドキュメントの編集中に整形式チェックと検証チェックが実行されます。（整形式のチェックに加え）検証を実行するにはDTD または XML スキーマがドキュメントに割り当てられている必要があります。エラーはエラーの存在するテキストを赤色で表示し、エラーの箇所を赤い感嘆符で表示することで表示されています。エラーのためにスマートフィックスを使用することができる場合、電球アイコンがエラーを生成するライン内に表示されます。アイコンにマウスをポイントすると、使用することのできるスマートフィックスのリストのポップアップが表示されます。適用するフィックスを選択します。詳細に関しては[ドキュメントの検証](271) を参照してください。

編集中に検証 モードは(i) [「XML | 編集中に検証」](847) メニューコマンド (ii) **編集中に検証** ツールバーボタン または(iii) [オプションダイアログの検証設定](1015)の *編集時* オプションによりオンとオフに切り替えることができます。

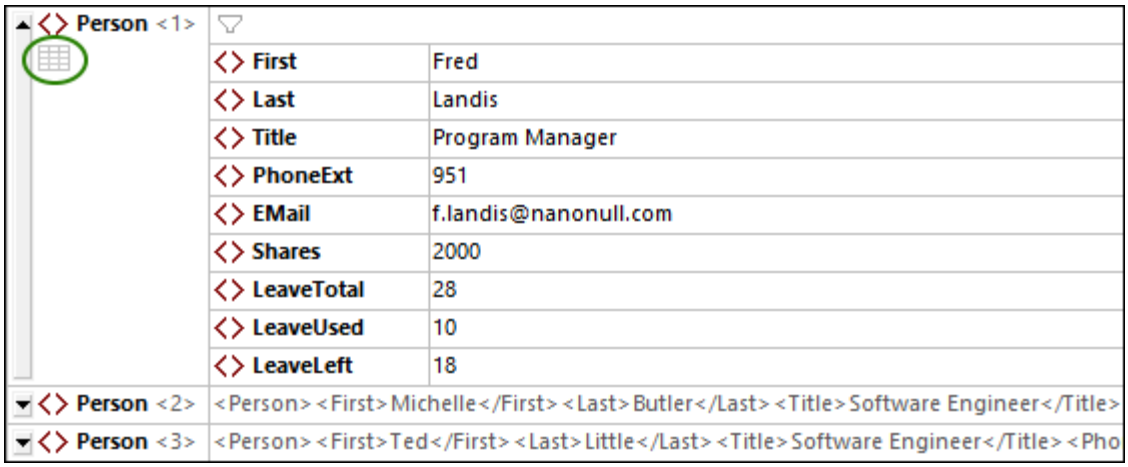
検索と置換

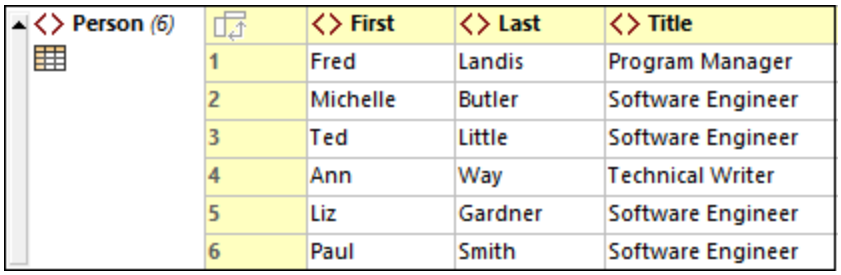
グリッドビュー内でテキストを検索し置き換えるために **編集** メニューの **検索** (Ctrl+F) と **置換** (Ctrl+H) コマンドを使用することができます。結果はオレンジ色でハイライトされ、含まれているセルもオレンジ色でハイライトされています。

テーブルとして表示する

繰り返し要素は*標準*のグリッドビューでドキュメント順に垂直下法で順番に表示されています(*左下のスクリーンショット参照*)。しかしながら、繰り返し要素を *テーブルの行* として表示すると追加編集機能が与えられます。下のスクリーンショットでは `Person` 要素は繰り返し要素です。左のスクリーンショットは標準のグリッドビューを表示しています。最初の `Person` 要素が展開されて表示され、続くインスタンスは折りたたまれて表示されます。右のスクリーンショットは繰り返し `Person` 要素をテーブルの行として表示しています。

テーブルビューに切り替えるには(左下のスクリーンショット内で緑色の丸で囲まれている) **テーブルモード** *アイコン をクリックします*。テーブルディスプレイに切り替えると、アイコンは色で表示されます (*右下のスクリーンショット*)。





テーブルビューはその行と列全体がテーブル内の他の列と行に相対的に操作可能になるユニークな編集の利点を提供します。これにより列の値上のテーブル行の並べ替えなどの操作が可能になります。例えば、右上のスクリーンショットでは、6個の `Person` 要素がグリッドビューツールバー 236 内の単一のコマンドにより `Last` 子要素をベースに並べ替えられることができます。これはXML ノードセットを並べ替えるための通常の方法であるXSLT 変換を実行するよりもシンプルです。

詳細に関しては、テーブル表示 250 のトピックを参照してください。

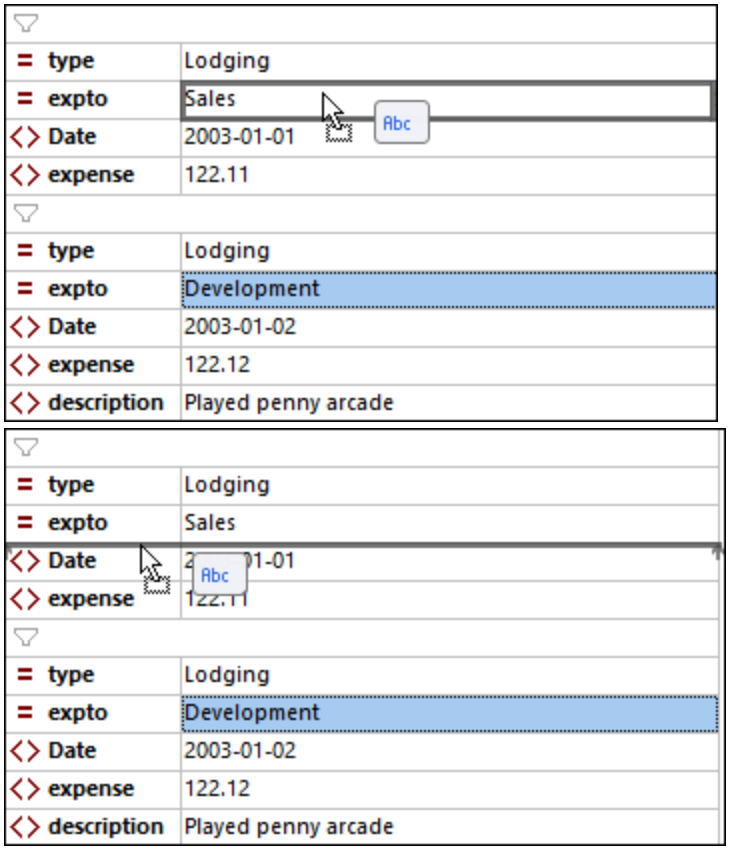
5.4.4 グリッドビュー内でのドラッグとドロップ

グリッドビューはJSON ドキュメントフラグメントをグリッドビューにXMLSpy 内、外部アプリケーション、またはW eb サイトで開かれているXML ドキュメントからドラッグすることのできる役に立つドラッグオーバーレイを提供します。

ターゲットノードにカーソルをポイントすると、ドラッグされている（アイテムの型と数量）だけでなくドロップされた際にどのようにアイテムが作成されるかについての情報を提供するドラッグオーバーレイが表示されます。例えば、下のスクリーンショットでは`Development` がドラッグ（移動）されています。アイテムが文字列であることをが`Abc` ポップアップにより示されています。

- （左側のスクリーンショットで示されている通り）値フィールドにポイントすると、フィールド全体がマークされ、マークされたフィールドの値がドラッグされたフィールドの値と置き換えられます。
- （右側のスクリーンショット内の`Date` などの）ノード名にポイントすると、値文字列をドロップするとツリー内にテキストノードを作成することを示す挿入ラインが表示されます。

メモ　　**Ctrl+Z** またはメニューコマンド **Edit | Undo**を使用して移動を元に戻すことができます。

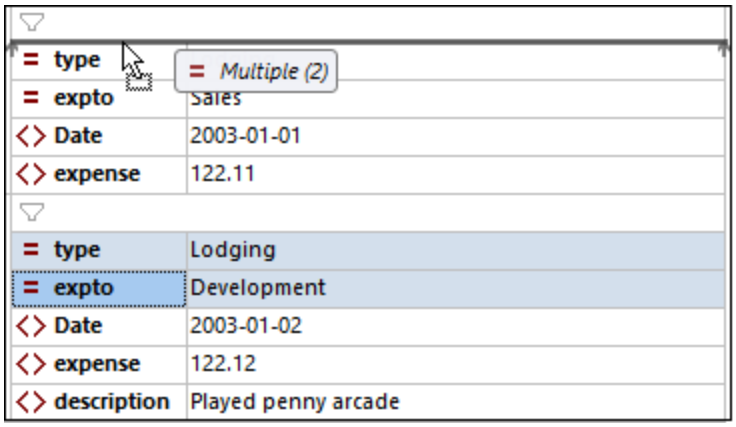


ドラッグオーバーレイ内に含まれている情報

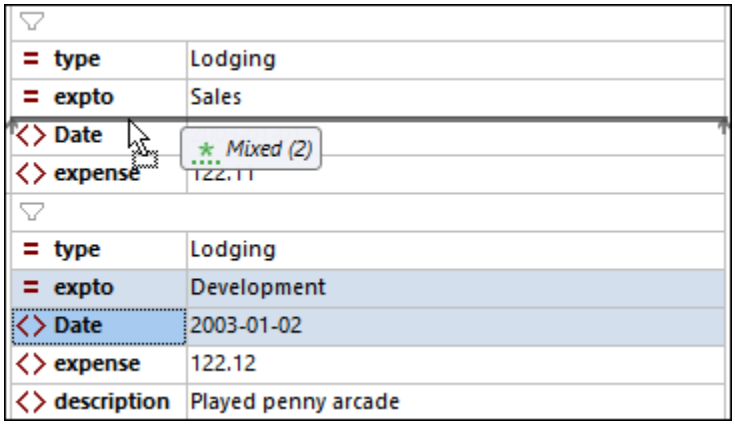
以下の種類のドラッグオーバーレイ情報が表示されます。

通常のグリッドビュー

- 値フィールドにドロップされた値はターゲットフィールド内の値を上書きするためにソース値を移動します（*左上のスクリーンショット参照*）。
- ツリーにノードとしてドロップされる値はテキストノードを作成します（*右上のスクリーンショット参照*）。
- 単純型の複数のノードのオーバーレイはドロップされたノード型とこれらのノードの数値を表示します。（値ではなく）ノードを選択するために、ノードの名前をクリックする必要があります。下のスクリーンショットでは、ドラッグオーバーレイ内の情報は2個の属性がドロップされることを示しています。

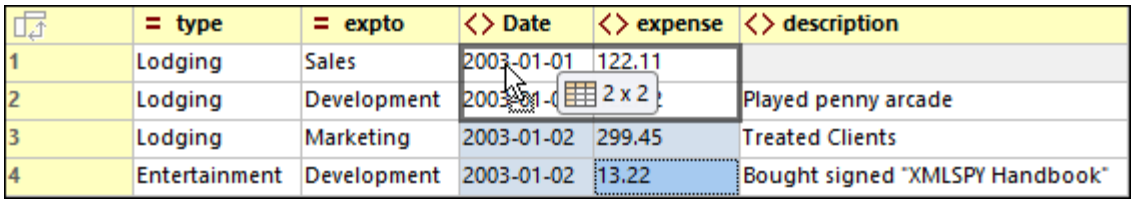


- 異なる型の複数のノードのためのオーバーレイはテキスト *混合* とクリップボード上のノードの数量を表示しています（*下のスクリーンショット参照*）。（値ではなく）ノードを選択するために、ノードの名前をクリックする必要があります。

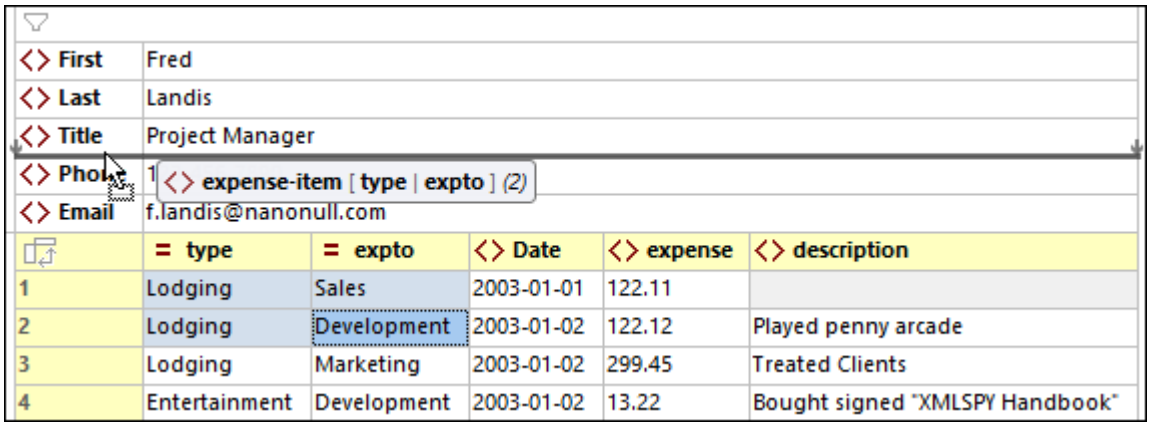


テーブルの表示

- 値がドラッグされると、選択されたセルの数値は 列 x 行 により示されています（*下のスクリーンショット参照*）。ドラッグされたセルの間トリックは対応するマトリックスを置き換えます。ターゲットセル（マトリックスをドロップするセル）はマトリックスの左のセルの値を受け取ります。他のドラッグされたセルがターゲットセルから右方向および下方向にフィルします。例えば、下のスクリーンショットでは青色のセルがドラッグされ最初の`Date`セルにドラッグされています。これによりマークされたセルが置き換えられます。ドラッグされたセルのマトリックスがテーブルの枠を超えると、列および/または行の適切な数量がすべてのドラッグされたセルに対応するように追加されます。この場合、拡張される枠は点線で表示されます。



- ターゲットがセルではなくノードツリーの場合、追加されるノードは(i)（角かっこに含まれる）追加されるノード、と(ii) 新規の要素のために追加される行の数量 と共に表示されます（*下のスクリーンショットを参照*）。

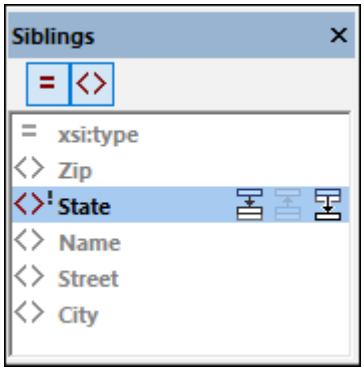


- （ヘッダーをクリックして）選択し、隣接する列ヘッダーにドロップして列を移動することができます。挿入ラインは移動された列が配置されるターゲット列のサイドを示しています。

5.4.5 グリッドビュー内の入力ヘルパー

グリッドビュー内のXMLドキュメントでは、3つの入力ヘルパーを使用することができます：兄弟、子、および値グリッドビュー内でセルが選択されていると、コンテキストを考慮したアイテムが各入力ヘルパーに表示されます。表示内容はドキュメントに割り当てられたスキーマ内で定義されているドキュメント構造とノード制約に依存します。

兄弟と子入力ヘルパー
子孫と子入力ヘルパーはそれぞれグリッドビュー内で選択されているノードの使用可能な子孫と子ノードを含んでいます。子孫入力ヘルパーの下のスクリーンショットを参照してください。既に追加されているノードは灰色表示されていますが、追加されていないノードは黒色で表示されています。必須な要素は感嘆符のマークが付いています。下のスクリーンショット内では必須 `State` 要素以外の全ての兄弟ノードが既にドキュメントに追加されています。



兄弟または子入力ヘルパー内でアイテムを選択すると右側にそれぞれグリッドビュー内で選択されているノードの直後、または直前にアイテムの挿入、またはすべての兄弟の後に（付加を）許可するアイコンが表示されます（*上のスクリーンショット参照*）。

値入力ヘルパー
グリッドビュー内で値を取ることができるノードが選択されており、そのノードがスキーマ内で定義済みの可能な値のセットを持っている場合これらの値は値入力ヘルパー内にリストされています。これらの値をダブルクリックして入力します。

5.4.6 テーブルの表示

テーブルの表示について

標準のグリッドビューでは、要素が繰り返されると、各要素インスタンスが表示されます。下のスクリーンショットでは Person 要素は繰り返し要素です。

Person <1>	
First	Fred
Last	Landis
Title	Program Manager
PhoneExt	951
EMail	f.landis@nanonull.com
Shares	2000
LeaveTotal	28
LeaveUsed	10
LeaveLeft	18
Person <2>	
First	Michelle
Last	Butler
Title	Software Engineer
PhoneExt	654
EMail	m.butler@nanonull.com
Shares	1500
LeaveTotal	19
LeaveUsed	9
LeaveLeft	10

上に表示される Person 要素などの繰り返し要素をテーブルとして表示することができます（*下のスクリーンショット*）。テーブル表示では、繰り返し要素の子ノードはテーブルの列を形成し、繰り返し要素のインスタンスはテーブルの行を形成します。テーブルビューに切り替えるには繰り返し要素の最初のインスタンス上に表示されている*（上のスクリーンショット内で緑色で囲まれている）* **テーブルモード** アイコンをクリックします。テーブルディスプレイに切り替えると **テーブルモード** アイコンは色で表示されます*（右のスクリーンショット）*。

Person (6)	First	Last	Title
1	Fred	Landis	Program Manager
2	Michelle	Butler	Software Engineer
3	Ted	Little	Software Engineer
4	Ann	Way	Technical Writer
5	Liz	Gardner	Software Engineer
6	Paul	Smith	Software Engineer

テーブルビューはその行と列全体がテーブル内の他の列と行に相対的に操作可能になるユニークな編集の利点を提供します。これにより列の値上の行の並べ替えなどの操作が可能になります。例えば、上のスクリーンショットでは、6個の Person 要素がGUI 操作により Last 子要素をベースに並べ替えられることができます。*（詳細に関しては下を参照してください）* このような操作はXML ノードセットを並べ替えるための通常の方法であるXSLT 変換を実行するよりもシンプルです。

メモ

- テーブル表示は同じ名前の要素のシーケンスに対してのみ適用することが可能です。
- テーブルの表示は単一の要素のために使用することができます。要素の**テーブルモード** アイコンは要素をクリックすると表示されます。
- オプションダイアログ内でテーブル表示の色を設定することができます: [フォントと色 | グリッドビュー | セルの色] 1031。

テーブル表示内のビューと編集のためのアイコン

下に表示されるアイコンはテーブル表示内で使用可能なアイコンで、ビューと編集機能を提供します。これらはテーブルセルおよび/またはグリッドビューツールバーで使用することができます。

	セルコンテンツに従いグリッド列の幅を最適化
/	色で表示されている場合 テーブルの表示 488 がオンに切り替えられています。それ以外の場合はオフです。表示するためにクリックします。
/	テーブルの左側のセル内のコマンドをトグルします。行を列に、または列を行に切り替えます。
/	色で表示されている場合、テーブル、オブジェクト、または配列のためのフィルター 492 が有効化されています。それ以外の場合フィルターは無効化されています。無効化/有効化するためにクリックします。式を編集するには、ダブルクリックします。
	セルをワードラップするためのトグルコマンド。選択されると、ワードラップがオンになります。デフォルトでは、アイテムのコンテンツのみがラップされます。アイテムの名前を追加でラップする場合、このオプションをアイコンのドロップダウンリストから選択します。
100%	グリッドビューのズームレベル
	テーブルビュー内で列ヘッダーが選択されると有効化されます。ボタンはテーブルの行を列のコンテンツの降順または昇順の順序にそれぞれ並べ替えます。

テーブル表示内の編集

テーブルの表示では以下で説明されている編集アクションを実行することができます。

テーブル行(テーブルの繰り返し要素の新規インスタンス)を追加します。

テーブルの繰り返し要素の他のインスタンスである新規の行を以下のように追加することができます:

1. 新規の行を追加する行の上または下の *行番号を含むセル*を右クリックします。
2. コマンド **後に挿入** (Ctrl+Enter) を選択して選択された行の下に行を追加、または **前に挿入** (Ctrl+Shift+Enter) を選択して選択された行の前に行を追加します。これらのコマンドは XML メニュー 812 とグリッドビューツールバー 236 で使用することができます。

新規の行は要素ノードとして作成されます。希望する場合ノードの型を変更することができます *(詳細に関しては下を参照してください)*。

兄弟または子要素をテーブルセルに追加する

テーブルセルがテーブル行の子要素を表示している場合、この子要素に以下の兄弟ノードまたは子ノードを与えることができます。テーブルセルを右クリックしてコマンド **付加** (Ctrl+Alt+A) または **この追加** (Ctrl+Alt+Enter) をそれぞれ選択します。これらのコマンドは XML メニュー 812 とグリッドビューツールバー 236 で使用することができます。新規の行は要素ノードとして作成されます。希望する場合ノードの型を変更することができます *(詳細に関しては下を参照してください)*。

要素内でセルをラップする
テーブルセルの周りに要素を作成することができます。セルとセルの親の間のレベルに新規の要素が作成されます。これを行うにはテーブルセルを右クリックしてコマンド **要素内でラップ**（Ctrl+Alt+W）を選択します。このコマンドはXML メニュー[812] とグリッドビューツールバー[236] で使用することができます。

テーブル列を追加します（テーブルの繰り返し要素のすべてのインスタンスの新規の子ノード）
テーブルの繰り返し要素のすべてのインスタンスの新規の子ノードである新規の列を以下のように追加することができます:

1. 列ヘッダーまたは列の空ではないセルを右クリックします。
2. コマンド **後に挿入**（Ctrl+Enter）を選択して選択された行の右側に列を挿入、または **前に挿入**（Ctrl+Shift+Enter）を選択して選択された行の左側に列を追加します。これらのコマンドはXML メニュー[812] とグリッドビューツールバー[236] で使用することができます。

新規の列は要素ノードとして作成されます。希望する場合ノードの型を変更することができます（*詳細に関しては下を参照してください*）。

列のノード型と名前を変更する
列のノード型を変更するには、列のノード型アイコンをクリックし、表示されるメニューから新規のノード型を選択します。このノードの型は繰り返し要素の全てのインスタンスのために変更されます。

列ヘッダーの名前を変更するには、名前をダブルクリックして編集します。この結果、このノードの名前は繰り返し要素の全てのインスタンス内で変更されます。

選択された列の値のテーブル行を並べ替える
自身の子ノード（列）の一つの相対的な値の上のテーブルの行を並べ替えることができます。例えば、アルファベット順に並べ替えられたテーブルの繰り返し要素を与えるために *LastName* 列をベースに行を並べ替えることができます。列を並べ替えるには、列ヘッダーを選択し、グリッドビューツールバー[236] コマンド **昇順で並べ替え** または **降順で並べ替え**をクリックします。これらのコマンドはXML メニュー[812] 内でも使用可能です。

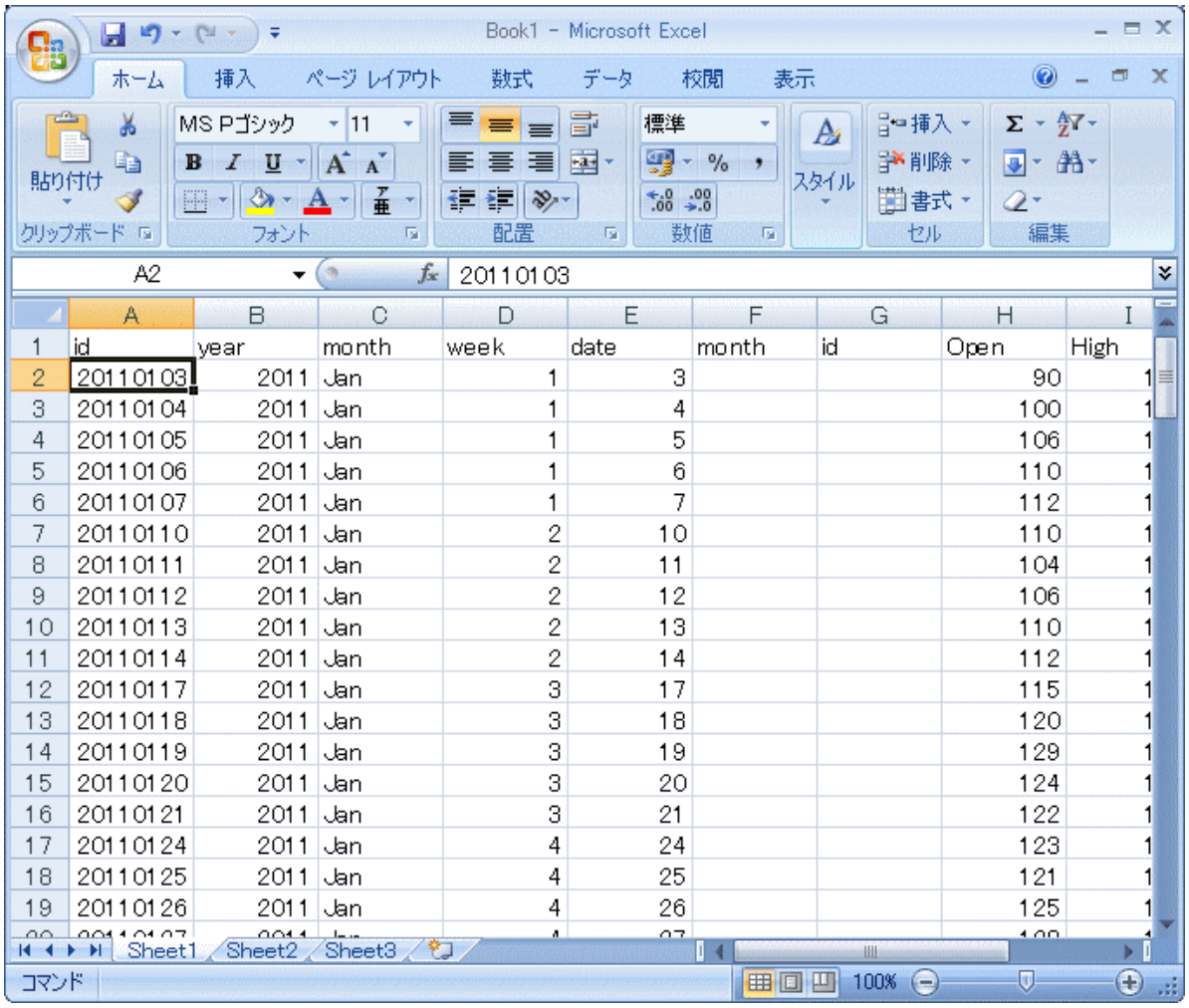
テーブル表示と外部アプリケーション

テーブル表示と（MS Excel などの）スプレッドシートアプリケーション間でデータを転送するためにテーブル構造を活用することができます。データの表示からデータを移動するには、以下を行います:

1. テーブルの表示内で、コピーするノードを選択します。セル自身、列ヘッダー、行ヘッダーまたはテーブル全体をクリックしてこれを行います。テーブル全体または列ヘッダーをクリックすると、列ヘッダーもコピーされます。それ以外の場合はコピーされません。（下のスクリーンショット内では、行 2 と行 3 が選択されています。）

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

2. コンテキスト コピー | タブで区切られたテキストとしてコピーする[764] を選択します。
3. データを直接スプレッドシートプログラムに張り付けます。



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

データの転送は両方向に行うことができます。スプレッドシートからテーブルの表示にデータをコピーすることができます。以下の手順を行ってください：

1. 外部アプリケーション内で範囲を選択して（**Ctrl+C** を使用して）クリップボードにコピーします
2. テーブルの表示内で単一のセルを選択します。
3. コピーされたデータを **Ctrl+V**を使用して貼り付けます。

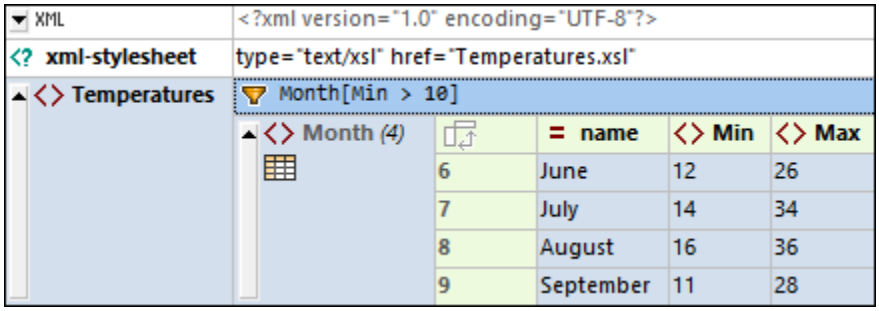
データは元の構造に対応し、テーブルビュー内で選択されたセルから開始する構造を持つテーブルに張り付けられます。以下の点に注意してください。

- データがテーブルビュー内のこれらのセルに既存の場合、新規のデータは元のデータを上書きします。
- 新規のデータに適用するために更なる行/または列が必要な場合、これらが作成されます。
- データは対応するセルにより表示される要素のコンテンツになります。

更に複雑なデータ転送タスクのために、XMLSpy のユニークな インポート/エクスポート 938 関数を使用してください。

5.4.7 フィルター

XML グリッドビュー内のフィルターを要素ノードに適用することができ、これによりそのノードの子孫をフィルターできるようになります。フィルターが適用可能な都度、灰色表示されたフィルターアイコンが表示されます。フィルターが一度定義されると、このアイコンは色が付けられて表示されます(*下のスクリーンショット参照*)。フィルターは XQuery 3.1 式を使用して定義されます。例えば、下のスクリーンショットではフィルターは **Temperatures** ノードは **10** より大きい値を持つ **Min** 子要素を持つ **Month** 要素のみを表示するように設定されています。(スクリーンショット内では対応する要素インスタンスのインデックス番号がリストされています。)



フィルターをセットアップするには、フィルターする要素を右クリックして **フィルター** を表示されるコンテキストメニューから選択し Xquery 式を入力して **入力**をクリックします。*フィルターセルはアイコン* [▼] により示されています(*上のスクリーンショットを参照*)。*フィルターアイコンをクリック*してフィルターを有効化/無効化することができます。

上のスクリーンショットで使用されている XML ドキュメントのリスティング

```
<?xml version="1.0" encoding="UTF-8"?>
<Temperatures>
   <Month name="January">
      <Min>-5</Min>
      <Max>3</Max>
   </Month>
   <Month name="February">
      <Min>-16</Min>
      <Max>1</Max>
   </Month>
   <Month name="March">
      <Min>-9</Min>
      <Max>7</Max>
   </Month>
   <Month name="April">
      <Min>2</Min>
      <Max>16</Max>
   </Month>
   <Month name="May">
      <Min>8</Min>
      <Max>21</Max>
   </Month>
   <Month name="June">
      <Min>12</Min>
      <Max>26</Max>
   </Month>
   <Month name="July">
      <Min>14</Min>
      <Max>34</Max>
```

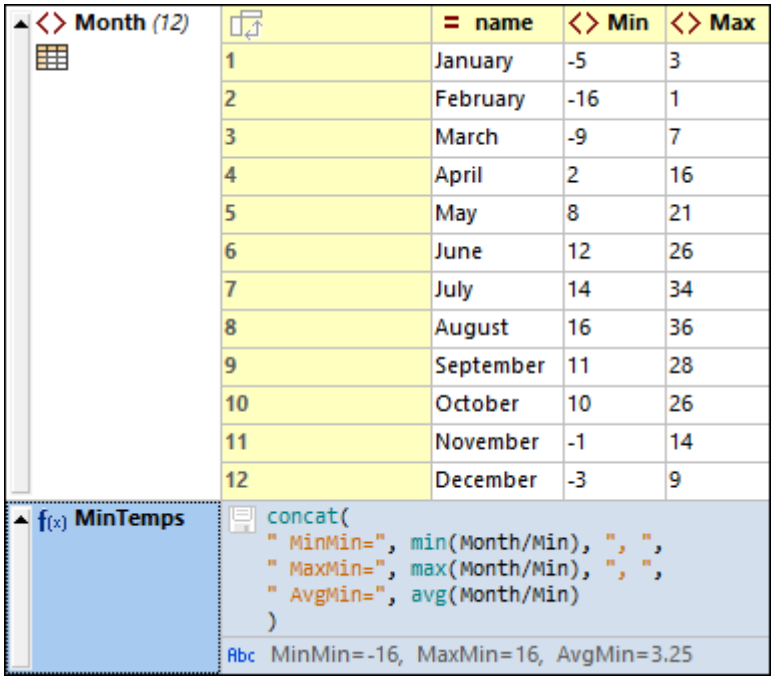
```
      </Month>
      <Month name="August">
         <Min>16</Min>
         <Max>36</Max>
      </Month>
      <Month name="September">
         <Min>11</Min>
         <Max>28</Max>
      </Month>
      <Month name="October">
         <Min>10</Min>
         <Max>26</Max>
      </Month>
      <Month name="November">
         <Min>-1</Min>
         <Max>14</Max>
      </Month>
      <Month name="December">
         <Min>-3</Min>
         <Max>9</Max>
      </Month>
   </Temperatures>
```

フィルターに関する以下の点に注意してください：

- フィルターは要素ノードのみに適用することが可能です。
- フィルターの XQuery 式のコンテキストノードは現在のノードです。上のスクリーンショットでは、例えば、コンテキストノードは **Temperatures** ノードです。
- フィルターをネストすることが可能です。ネストされたフィルターは親フィルターのフィルターされたコンテンツに適用することができます。
- 式内に新規のラインを追加するには、「**Ctrl+Enter**」を押します。更に読みやすいようにするために式を複数行に表示するためにこの機能は役に立ちます。
- フィルターされたコンテンツは視覚的な表示のためだけのものです。実際のコンテンツは変更されずそのままです。
- 各フィルターは個別に実行され他のフィルターまたはフィルター 255 により影響されません。
- フィルターは XML ドキュメント内で保管されませんが、次回の使用のために特別なメタデータとして保管することができます。フィルターが保管されると、グリッドビュー内でドキュメントが再度開かれると自動的に適用されます。ツール | オプション | ビュー | グリッドビュー設定 1025 に移動して、フィルターを常に自動的に保管、フィルターをリクエストにより保管、またはフィルターを保管しないことを指定します。

5.4.8 フォーミュラ

結果を計算またはドキュメント内に保管されるノードセットを生成するために XML グリッドビュー内のフォーミュラは XQuery 3.1 式を使用します。フォーミュラは XQuery 3.1 式を使用して定義されます。例えば、下のスクリーンショットでは最低、最高、および全ての最低気温のセットの平均気温を計算する **MinTemps** という名前のフォーミュラが作成されています。



Month (12)

	name	Min	Max
1	January	-5	3
2	February	-16	1
3	March	-9	7
4	April	2	16
5	May	8	21
6	June	12	26
7	July	14	34
8	August	16	36
9	September	11	28
10	October	10	26
11	November	-1	14
12	December	-3	9

MinTemps

```
concat(
  " MinMin=", min(Month/Min), ", ",
  " MaxMin=", max(Month/Min), ", ",
  " AvgMin=", avg(Month/Min)
  )
```

Abc MinMin=-16, MaxMin=16, AvgMin=3.25

フォーミュラを作成するには、以下を行います:

1. フォーミュラを表示する箇所に新規のノードを追加します。
2. ノードはデフォルトで要素として作成されます。ノードの型を243フォーミュラに変更します。
3. `f(x)` アイコンを含むセルをダブルクリックし、フォーミュラの名前を入力します *(上のスクリーンショットを参照)*。
4. このアイコンをクリックして、フォーミュラの出力をドキュメントに保存することができます。
5. 式のセルをダブルクリックして式内に入力し「Enter」をクリックします。

上のスクリーンショットで使用されている XML ドキュメントのリスティング

```
<?xml version="1.0" encoding="UTF-8"?>
<Temperatures>
   <Month name="January">
      <Min>-5</Min>
      <Max>3</Max>
   </Month>
   <Month name="February">
      <Min>-16</Min>
      <Max>1</Max>
   </Month>
   <Month name="March">
      <Min>-9</Min>
      <Max>7</Max>
   </Month>
   <Month name="April">
      <Min>2</Min>
      <Max>16</Max>
   </Month>
   <Month name="May">
      <Min>8</Min>
      <Max>21</Max>
```


(C) 2015-2021 Altova GmbH

```
    </Month>
    <Month name="June">
       <Min>12</Min>
       <Max>26</Max>
    </Month>
    <Month name="July">
       <Min>14</Min>
       <Max>34</Max>
    </Month>
    <Month name="August">
       <Min>16</Min>
       <Max>36</Max>
    </Month>
    <Month name="September">
       <Min>11</Min>
       <Max>28</Max>
    </Month>
    <Month name="October">
       <Min>10</Min>
       <Max>26</Max>
    </Month>
    <Month name="November">
       <Min>-1</Min>
       <Max>14</Max>
    </Month>
    <Month name="December">
       <Min>-3</Min>
       <Max>9</Max>
    </Month>
</Temperatures>
```

上のスクリーンショットで使用されているフォーミュラ式のリスティング

```
concat(
" MinMin=", min(//Month/Min), ", ",
" MaxMin=", max(//Month/Min), ", ",
" AvgMin=", avg(//Month/Min)
)
```

以下の点に注意してください：

- フォーミュラのXQuery 式のコンテキストノードはフォーミュラノードの親ノードです。
- 式内に新規のラインを追加するには、「Ctrl+Enter」を押します。更に読みやすいようにするために式を複数行に表示するためにこの機能は役に立ちます。
- （フォーミュラを定義する式である）フォーミュラは（アプリケーション全体メタデータファイル内に存在する）ドキュメントのメタデータ内に保管されています。フォーミュラはXML ファイル内に処理命令としても保存されます。
- フォーミュラにより生成される結果はフォーミュラのXQuery 式の下のセル内に表示され、ドキュメントのメタデータ内に保管されます。結果はドキュメント内でも保管することができます。XQuery 式の横のディスクアイコンをクリックしてこれを行います。アイコンがクリックされているかに関わらず、フォーミュラの結果は計算されドキュメントのメタデータに保管されます。
- フォーミュラの結果が計算の場合、フォーミュラに割り当てられた名前を持つ要素のコンテンツとして保管されます。例えば、上のスクリーンショットに表示されている**MinTemps** フォーミュラでは、結果は**MinTemps** という名前が付けられている要素内に保管されます。
- ツール | オプション | ビュー | グリッドビュー設定 1025 に移動して以下を行います：(i) メタデータ内にフォーミュラを常に、リクエストに応じて保管するか、または決して保管しないかを指定する。(ii) ドキュメント内でフォーミュラを保管するか指定する。

テーブル内のフォーミュラ

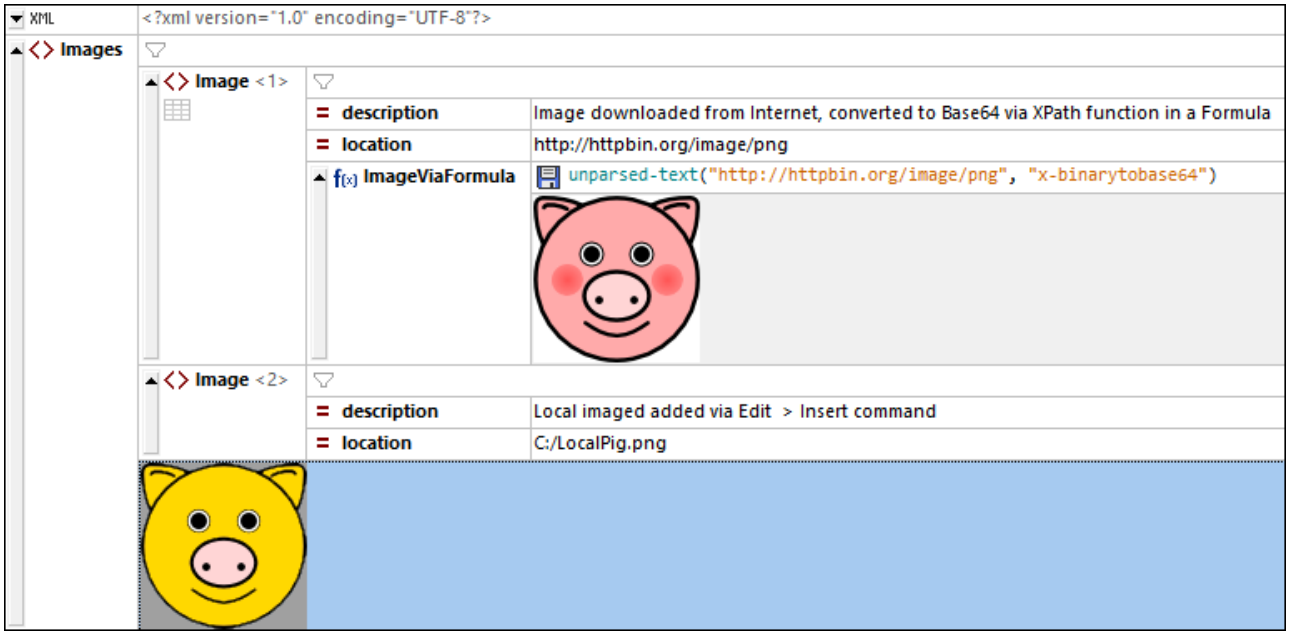
(テーブルディスプレイ 488) 内のテーブル列のすべてのセルに同じフォーミュラが含まれている場合、フォーミュラは列のヘッダー内で一度のみ表示されます *(下のスクリーンショットを参照)*。フォーミュラ計算の*結果*はしかしながら対応するセル内に表示されます。列ヘッダー内のフォーミュラはグリッドビュー表示です。(テキストビュー参照) XML ドキュメント内で、フォーミュラは各テーブル行アイテムのために繰り返されています。

Month (12)	name	Min	Max	MidRange (Min + Max) div 2
1	January	-5	3	# -1
2	February	-16	1	# -7.5
3	March	-9	7	# -1
4	April	2	16	# 9
5	May	8	21	# 14.5
6	June	12	26	# 19
7	July	14	34	# 24
8	August	16	36	# 26
9	September	11	28	# 19.5
10	October	10	26	# 18
11	November	-1	14	# 6.5
12	December	-3	9	# 3

セルの単一フォーミュラが異なる場合でも、各フォーミュラは対応するセル内で表示されます。フォーミュラがヘッダーに表示されるようにテーブル列のすべてのフォーミュラが同じの場合、および個別のセルのために異なるフォーミュラを作成する場合、テーブルの表示をオフに切り替えて、そのセルのフォーミュラを編集します。テーブルディスプレイに切り替えると、フォーミュラは(全てのセルのために)個別のセルのために表示されます。

5.4.9 イメージ

イメージを直接グリッドビューで表示することができます *(下のスクリーンショット参照)*。これを行うには、イメージはBase64-エンコードで挿入される必要があります。



グリッドビューセルにBase64-エンコードイメージを挿入する方法が2通りあります:

- 型 *Formula* であるノードを作成します。フォーミュラ式セル内で *(上のスクリーンショット参照)* イメージをBase64 エンコードに変換する以下のXPath 式を入力します: `unparsed-text("<Image-URL>", "x-binarytobase64")`。XPath 関数 `unparsed-text` はイメージをBase64 エンコードに変換します。イメージはフォーミュラ式を含むセルの下に表示されます。
- イメージを追加するセル内にカーソルをポイントします。例えば、上のスクリーンショットでは、`Image` 要素を選択することによりイメージは2番目の`Image` 要素のコンテンツとして追加されます。[**編集 | 挿入 | エンコードされた外部ファイル**]を選択します。表示されるダイアログ内: (i) イメージへのパス(ローカルまたはインターネット)を入力します(ii) *Base 64* を選択します。(iii) *テキストの作成*を選択します。イメージはBase64 エンコードに変換され、エンコードされたテキストはセル内に挿入されます。しかしながら、エンコードではなくセル内のイメージが表示されます *(上のスクリーンショット参照)*。

URL を相対的パスを含む以下の書式で提供することができます:

- `http://httpbin.org/image/png`
- `file:///c:/LocalPig.png`
- `C:/LocalPig.png`
- `LocalPig.png`

一般的に使用されるイメージフォーマットはサポートされています。これらはPNG、JPEG、BMP、およびアニメーション GIF を含みます。SVG はXMLドキュメントとして読み取られます。グリッドビューではイメージは`<svg>` 要素の最後の子として表示されます。

5.4.10 チャート

`altovaext:chart` という名前のAltova XPath/XQuery 拡張子を使用してXML グリッドビュー内でチャートを作成することができます(*下のスクリーンショットを参照*)。拡張子は下で説明されています。これに関してはチャート関数のセクション内で他のチャート共に説明されています。下のスクリーンショットで表示されている`altovaext:chart` 拡張子は グリッドビューフォーミュラ255 内で定義されているXQuery `Let` 式内で使用されています。チャートはフォーミュラの下でイメージとして表示されています。(下で与えられているXMLドキュメントリスティングとXquery 式を使用しチャート関数を試すことができます。)

XML <?xml version="1.0" encoding="UTF-8"?>

xml-stylesheet type="text/xsl" href="Temperatures.xsl"

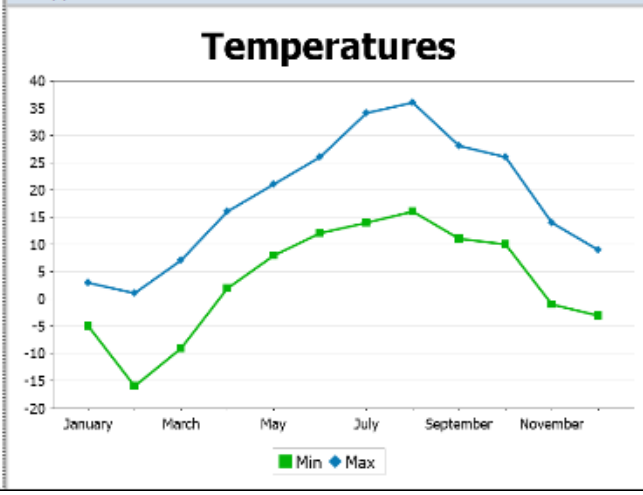
Temperatures

Month (12)

	name	Min	Max
1	January	-5	3
2	February	-16	1
3	March	-9	7
4	April	2	16
5	May	8	21
6	June	12	26
7	July	14	34
8	August	16	36
9	September	11	28
10	October	10	26
11	November	-1	14
12	December	-3	9

f(x) new_elem

```
let $months := //Month return
altovaext:chart(map{ "title": "Temperatures", "kind": "LineChart" },
(
    (:name, X-axis,        Y-axis :)
    ['Min', $months/@name, $months/Min],
    ['Max', $months/@name, $months/Max]
))
```



上のスクリーンショットで使用されている XML ドキュメントのリスティング

```
<?xml version="1.0" encoding="UTF-8"?>
<Temperatures>
   <Month name="January">
      <Min>-5</Min>
      <Max>3</Max>
   </Month>
   <Month name="February">
      <Min>-16</Min>
      <Max>1</Max>
```

```
    </Month>
    <Month name="March">
       <Min>-9</Min>
       <Max>7</Max>
    </Month>
    <Month name="April">
       <Min>2</Min>
       <Max>16</Max>
    </Month>
    <Month name="May">
       <Min>8</Min>
       <Max>21</Max>
    </Month>
    <Month name="June">
       <Min>12</Min>
       <Max>26</Max>
    </Month>
    <Month name="July">
       <Min>14</Min>
       <Max>34</Max>
    </Month>
    <Month name="August">
       <Min>16</Min>
       <Max>36</Max>
    </Month>
    <Month name="September">
       <Min>11</Min>
       <Max>28</Max>
    </Month>
    <Month name="October">
       <Min>10</Min>
       <Max>26</Max>
    </Month>
    <Month name="November">
       <Min>-1</Min>
       <Max>14</Max>
    </Month>
    <Month name="December">
       <Min>-3</Min>
       <Max>9</Max>
    </Month>
</Temperatures>
```

上のスクリーンショット内で表示されているチャートを生成するための XQuery 式のリスティング

```
let $months := //Month return
altovaext:chart(map{ "title":"Temperatures", "kind":"LineChart" },
(
    (:name, X-axis,       Y-axis   :)
    ['Min', $months/@name, $months/Min],
    ['Max', $months/@name, $months/Max]
))
```

Altova チャート拡張子の使用

- チャート拡張関数 `altovaext:chart` は名前空間プレフィックス `altovaext:` を使用する必要があります。
- `altovaext:chart` は2個の引数を取ります: (I) チャート構成情報と(ii) チャートデータ系列情報。
- チャート構成情報は `altovaext:chart` の最初の引数です。4個のキー—値ペアの順序指定されていない系列です。これらのペアは以下のために存在します: (i) チャートのタイトル(キーは `title`)、(ii) 円グラフ、折れ線グラフなどのチャートの種類(キーは `kind` 使用可能な種類に関してはチャート関数 を参照してください。)、(iii) ピクセルで表示されたチャートの幅(整数のみ。キーは `width` です)。および(iv) ピクセルで表示されたチャートの高さ(整数のみ。キーは `height` です)。幅または高さの値のどちらかが、または両方の値が与えられていない場合、不足する値または値はデータを基にして自動的に計算されます。
- チャートデータ系列は `altovaext:chart` の2番目の引数です。各データ系列はサイズ3の配列です: (I) 系列の名前 (ii) x-軸の値 (ii) y-軸の値(例えば、上のサンプルのように各系列がラインを表示する場合など) 複数の系列を作成する場合は複数の配列のシーケンスを作成します。上のサンプルには最低、最高、および平均機能のための3個のデータ系列が存在します。XとY 軸のためのデータはすべての `Month` 要素のシーケンスから参照されます。

詳細に関しては、チャート関数のセクションを参照してください。

5.4.11 グリッドビュー内のコンテキストメニュー

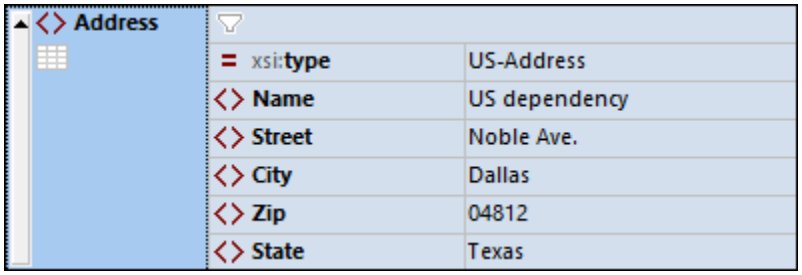
グリッドビュー内のセルを右クリックするとセルに関連したコンテンツを編集し、表示を変更するためのコマンドを提供するコンテキストメニュー(*下のスクリーンショット*)が表示されます。コンテキストメニューにキーボードの **メニュー** キーを押すことによりアクセスすることができます。コンテキストメニューのコマンドは下で説明されています。

型
型 コマンドにマウスをポイントするとコンポーネントの型 243 を選択することのできる表示されるサブメニューを表示します。キーボードショートカット **Ctrl+Menu** を使用して **Type** サブメニューにアクセスすることが可能です。

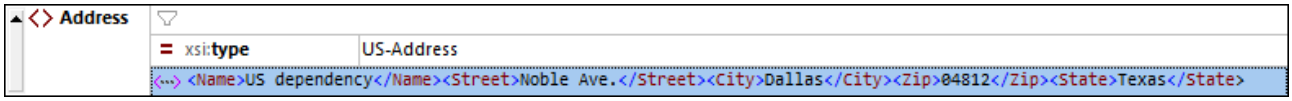
子を前/後に挿入、子の付加、子の追加
「**挿入**」と「**追加**」コマンドにより同じレベルでアイテム（デフォルトで要素）を追加することができます。**後に挿入** と **前に挿入**の場合新規のアイテムがそれぞれ選択されたアイテムの前後に追加されます。**付加** の場合、選択されたアイテムの最後の兄弟として追加されます。「**子の追加**」コマンドは新規のアイテム（デフォルトで要素）を子として追加します。名前セルをダブルクリックして編集し新規に追加された要素の名前を変更します。要素の（名前の左側の）アイコンをクリックして、必要とするノード型を選択しノード型を変更します。

要素内のラップ
（名前をダブルクリックして編集して）選択されたアイテムにデフォルトの要素を持つ親要素が与えられます。

生テキストとして編集
選択されたアイテムのテキストコンテンツを生テキストとして編集することが可能になります。HTML コードのような複雑なコンテンツを編集する場合などにこの機能は役に立ちます。例えば、下のスクリーンショットではグリッドビュー内の`Address` 要素を表示しています。



Address 要素を生テキストを表示する場合（*下のスクリーンショット*），グリッド構造はテキストビュー内同様に編集可能な生テキストを含む単一アイテムに変換されます。



要素空白文字
要素ノードで使用が可能です。**xml:space** 属性を要素のマークアップに追加し、コマンドのサブメニューから選択する値をこの属性に与えます。

- しかしながら *省略* 値を選択すると空白を正規化する効果がある **xml:space** 属性は追加されません。
- *保管* 値はすべての空白をそのままに保管し、その要素のために整形出力 123 をオフに切り替えます。保管された空白はグリッドビューでは省略記号のアイコンで表示されています。
- *デフォルト* 値はスキーマ内のその要素のために指定された値を取り、XML ドキュメント内の祖先要素から継承されている **xml:space** 値をオーバーライドするために役に立つ場合があります。

上/下/左/右への移動
コンポーネントを現在の位置からグリッド内で上下左右に移動することが可能になり、対応するコマンドが有効化されます。移動を実行するために対応するコマンド選択してください。

コピー
これらのコマンドは現在の選択項目をコピーするために使用されます。そのコンポーネントに許可されているオプションのみを使用することができます。

コマンド	説明
XML テキストとしてコピーする	現在の選択範囲はXML マークアップとしてシリアル化されています。
タブで区切られたテキストとしてコピー	現在のテーブル選択項目がTSV（タブ区切り値）としてシリアル化されます。
イメージとしてコピーする	現在のイメージセルがイメージとしてコピーされます

以下の点に注意してください：

- *イメージとしてコピー*は選択されたイメージのBase64 エンコード文字列をコピーします。（例えば、テーブルの表示内の他のセルなど）Base64-エンコード済み文字列がイメージとして表示されるドキュメントに文字列が張り付けられると文字列は標示されます。それ以外の場合は文字列として解析されます。
- テキストをセルに挿入するには、テキストをコピーしてセルに張り付けます。
- 例えばイメージなどをファイルからインポートするには、[編集 | 挿入 | エンコードされた外部ファイル] 768 コマンドを使用します。Base64 エンコード文字列としてイメージファイルを挿入し、この文字列をJSON グリッドビューで表示します。インポートのためにサポートされているイメージファイルフォーマットは以下の通りです：PNG、JPEG、BMP、GIF、TIFF。

XPath のコピー
選択されたノードをロケートするドキュメントルートから開始するPath 3.1 ロケーター式をクリップボードにコピーします。

XPointer のコピー
このコマンドは選択されたノードをロケートするXPointer 式をクリップボードにコピーします。詳細に関しては**XPointer/JSON-Pointer のコピー**[768] を参照してください。

フィルター、フォーカスするフィルター
フィルター コマンドは切り替えコマンドです。選択された要素にフィルターを追加またはフィルターを無効化にします。このコマンドを使用してフィルターを追加後、フィルター式を入力する必要があることに注意してください。それ以外の場合フィルターは自動的に削除されます。フィルターが追加された後、コマンドをもう一度選択またはフィルターをクリックして無効化することができます。フィルターを削除するにはツールバー内で *全てのフィルターを削除する* をクリックします。

フォーカスするフィルター コマンドは (i) 選択されたノードをロケートするXPath 式を含むルート要素上にフィルターを作成します。そして (ii) 選択されたノード (とその子孫) を直接ルート要素の下で表示するためにグリッドビューをフィルターします。選択されたノードの表示のみに集中する場合この機能は役にたちます。フィルターを削除するには、ルート要素のコンテンツセル内のフィルターアイコンをクリックします。

フルに展開する
このコマンドは選択されているコンポーネントまたはその子孫が折りたたまれている場合に有効化されます。コンポーネントと子孫コンポーネントが展開されます。

選択されていない範囲を折りたたむ
このコマンドは選択されているコンポーネントとその祖先コンポーネント以外のすべてのコンポーネントを折りたたみます。

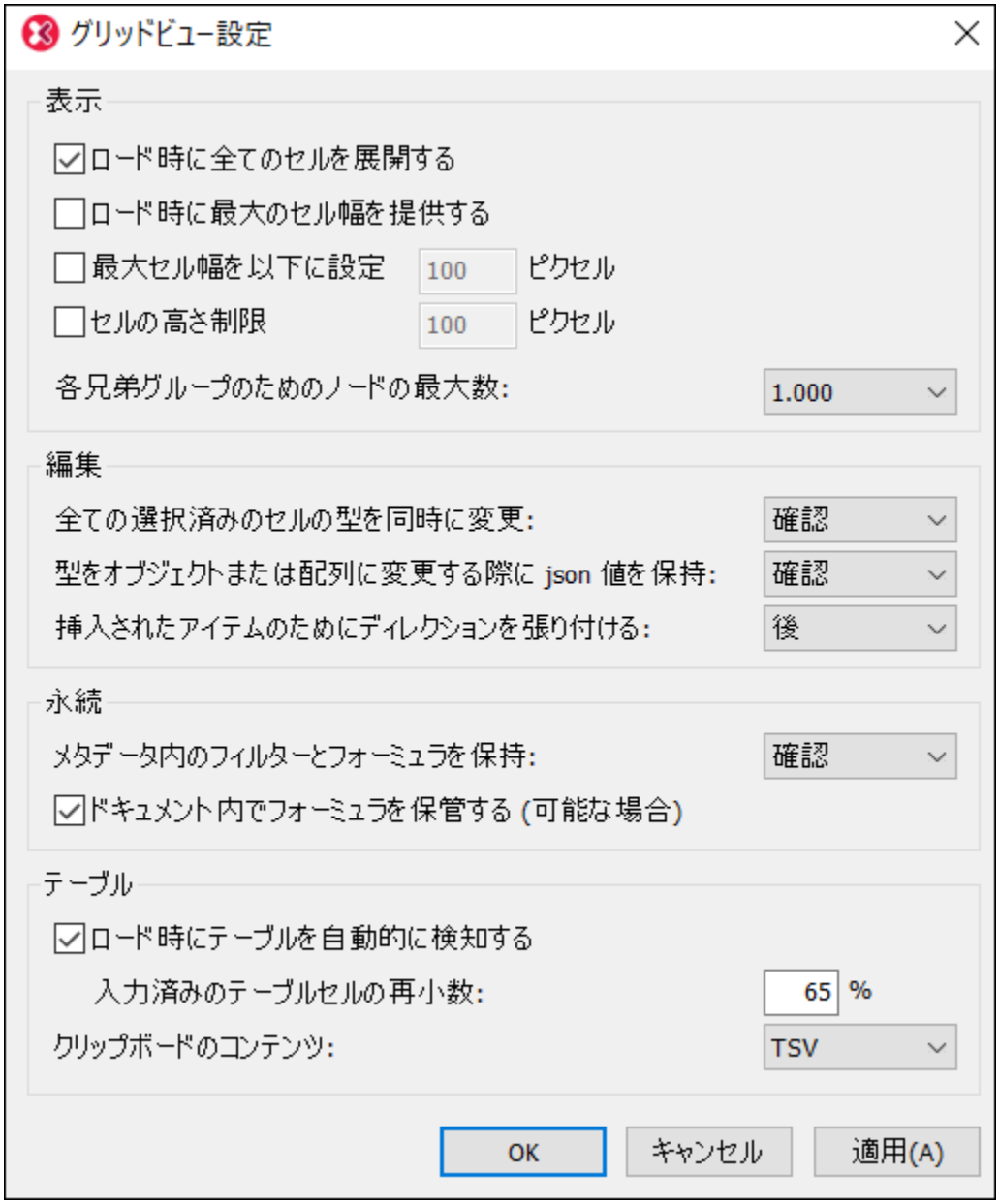
テキストの向き
このコマンドはテキストを含むノードのために使用することができ、読み取りの順序をセルの左側方向または右側方向に切り替えます。この機能はアラブ語とヘブライ語などの言語を使用する際に役に立ちます。

5.4.12 グリッドビュー設定

グリッドビューの設定はグリッドビュー設定ダイアログ (*下のスクリーンショット*) は[グリッドビューツールバー](236) 内の設定アイコンを使用してアクセスすることができます。

	[グリッドビューツールバー](236) 内のグリッドビュー設定アイコン

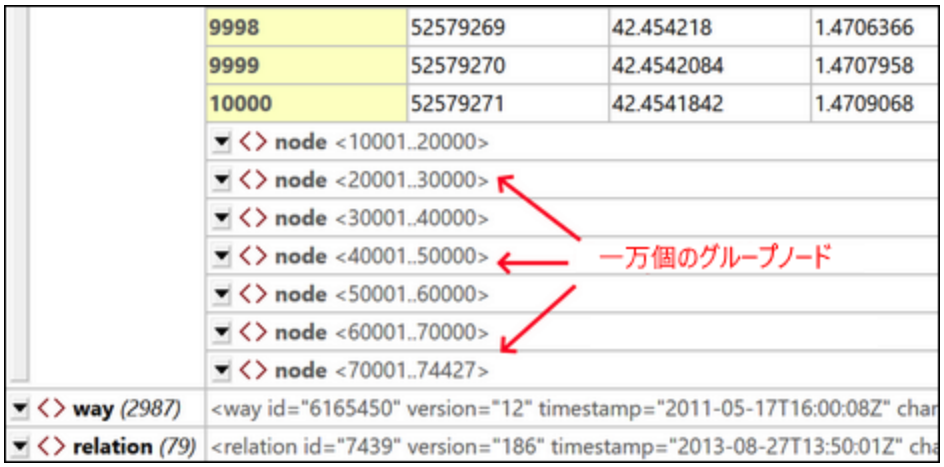
グリッドビューの設定は以下で説明されています。全てのドキュメント (XML、JSON、DTD) のグリッドビューに適用することができることに注意してください。



表示

表示 セクション内のチェックボックスは自明です。以下は更に説明が必要な場合のメモです。

- ロードする際に全てのセルが展開されていない場合ルートノードとそのすべての子孫は折りたたまれています。ドキュメントを更に深く移動するためには各ノードを展開する必要があります。
- 最適な幅がオンに切り替えられている場合グリッドの幅全体が表示されます。これを達成するには、セルの一部のテキストがラップされます。
- 兄弟ノードは100、1000 または10000 ノードの兄弟グループで表示することができます（*下のスクリーンショットを参照してください*）。多数の兄弟ノードを持つファイル内でスペースの表示を省略する場合



編集
表示 セクション内のチェックボックスは自明です。以下は更に説明が必要な場合のメモです。

- 複数の選択されたセルの型を変更する場合、設定のアクションを継続するかを決定する以下のオプションが与えられます: (ユーザーにより決定される) *常に*、*なし* または *確認* 。
- JSON 型を動的な型からオブジェクトまたは列挙に変更する場合、設定のアクションを継続するかを決定するかを決定する以下のオプションが与えられます: (i) *確認* (動的な型の値が名前を持たない子 `key:value` ペアの値として保持される、または破棄されるかを確認) (ii) *常に* (名前を持たない子 `key:value` ペア内の値を保持する) (iii) *なし* (値を保持する。すなわち値を破棄する)。
- *ディレクションの貼り付け* オプションはクリップボード内の選択範囲が選択されたセルの上または下に張り付けられるかを決定します。

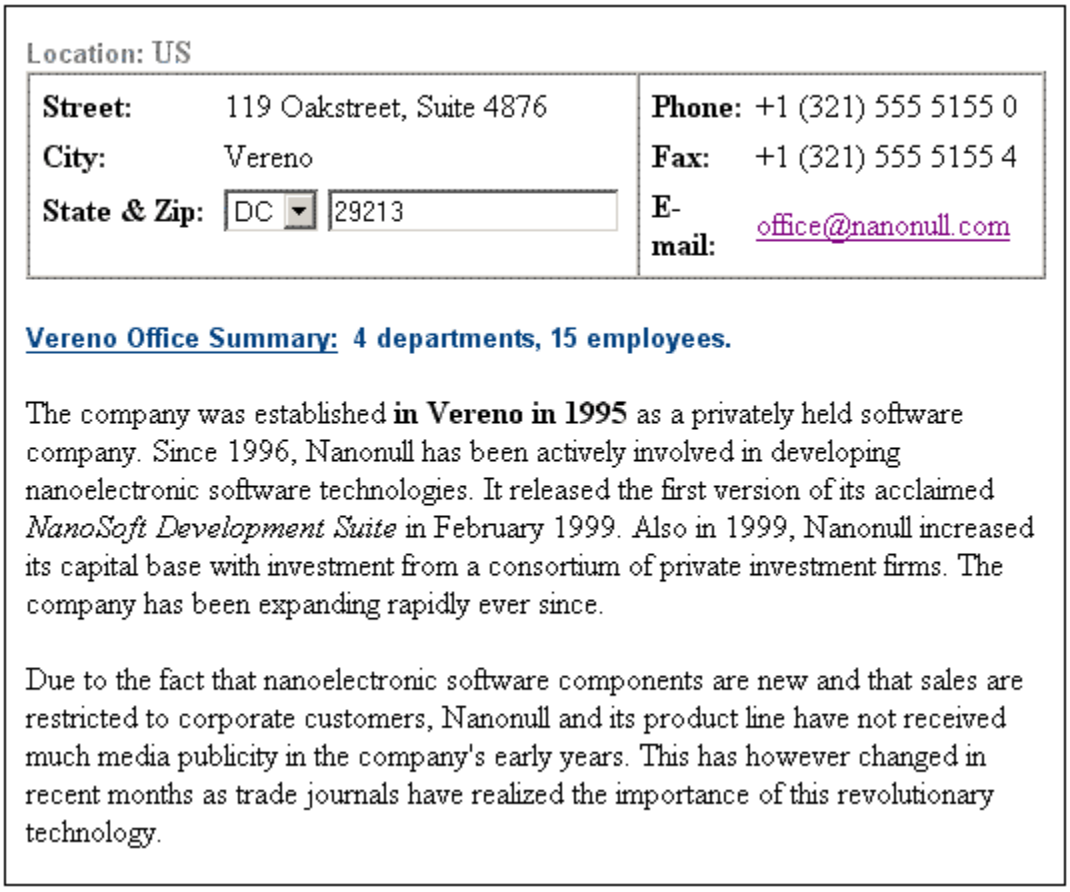
永続
フィルターとフォーミュラを XML と JSON ドキュメントのメタデータ内に保管することができます。フォーミュラは XML ドキュメント内の処理命令として、そして JSON ドキュメント内のコメントとして保管されます。これらの設定はフィルターとフォーミュラの保管の方法を指定します。フォーミュラの場合 *可能な場合* という用語はフォーミュラが実際にすべての JSON ドキュメント (JSON5 と JSONC ドキュメント内でのみ) のコメントとして保存できないことを意味します 。

テーブル
以下の設定を使用することができます:

- 設定がロード時に JSON グリッドビューテーブルを自動的に検知する場合、テーブルをテーブルとして検知するために評価する記入されたテーブルセルの最小のパーセントを選択することができます。記入されたテーブルセルの数値がこのレベルを越さない場合、構造はそれぞれが下に表示されている繰り返し要素を持つ通常のグリッドとして表示されます。JSON グリッドビューのみにこの設定は適用されることに注意してください。XML グリッドビュー内では、テーブルセルの記入されている率に関わらずテーブルは自動的に検知されます。
- クリップボードコンテンツがタブにより区切られた値 (TSV) または (ドキュメントの型に従い) XML/JSON として保管するかを選択することが可能です。これはとても役に立つ機能です: クリップボードからテーブルを他のドキュメントに張り付ける場合、この設定により TSV としてまたはマークアップとしてテーブルがコピーされるかを選択できるようになります。(違いを確認するには、テーブルをそれぞれのフォーマットでクリップボードにコピーしテキストエディターにテーブルを張り付けてください。)

5.5 Authentic ビュー内の XML

Authentic ビューによりユーザーはテキストドキュメントのように XML ドキュメントを編集することができます（*下のスクリーンショット*）。XML マークアップとすべての他の非コンテンツテキストはドキュメントの編集者に対して非表示になります。 cこの機能はドキュメントのコンテンツに集中しつつ有効なドキュメントの作成を可能にすることにより XML に慣れていないユーザーにとり役に立ちます。



StyleVision Power Stylesheet（SPS）がXML ドキュメントに割り当てられている場合ドキュメントのAuthentic ビューは有効化されます。XML ドキュメントがベースにされているスキーマソースにSPS はベースされており XML ドキュメントの構造を定義します。SPS は Authentic ビューのドキュメントのレイアウトと書式設定を定義します。例えば、上のスクリーンショットに表示されるドキュメント内では、以下のAuthentic 書式設定と編集機能が使用されています:

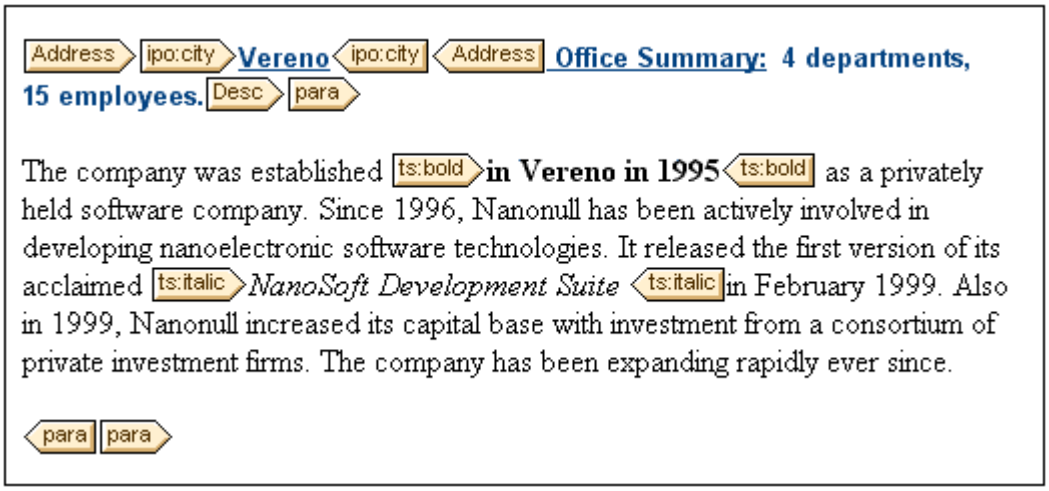
- パラグラフと他のブロック書式設定
- テーブル構造
- 色とフォントフェースなどのテキストの書式設定
- （State とZip フィールドにある）コンボボックスにより、妥当な選択肢の中から値を選択することができます。選択肢は上で記述されているように、スキーマ内にある 列挙 から取得されます。
- ドキュメント内のデータから計算を行ない表示することができます。上の例では、Office Summary の情報はユーザーによって入力されたものではなく、ドキュメント内のデータから計算して得られたものです。

SPS はXML ドキュメントをAuthentic ビューにて閲覧、そして XML から(HTML、PDF、RTF、そしてW ord 2007 といった）標準出力を生成するためのファイルです。SPS ファイルはAltova StyleVision により作成されます。

ドキュメント構造の編集

位置を選択し、(要素や属性)入力ヘルパー、またはコンテキストメニューから目的のノードを選択することで、妥当なノードをいつでもドキュメントに挿入することができます。カーソルの位置を変更することで、選択できるノードがその場所において兄弟または子要素として妥当性を失わずに挿入することができるものに制限されます。例えば、カーソルがパラグラフ内に位置している場合、スキーまでそれが許されていれば、新たなパラグラフを追加することができます。

XML ドキュメントの構造を Authentic ビューで編集する場合、ドキュメントのマークアップを参照できれば便利です。メニューオプションの **Authentic | 大きなマークアップを表示** (または対応するツールバーアイコン)を使用することで(*以下のスクリーンショットにあるように*)マークアップをタグとして表示することができます。



コンテンツの編集

ドキュメントのノード内に入力することでコンテンツの作成や編集を行うことができます。エンティティや CDATA セクションはコンテキストメニュー(そしてエンティティはエンティティ入力ヘルパーからも)から追加することができます。

Authentic ビューにおける編集の詳細

Authentic ビューにおける編集に関する詳細については、Authentic ビューセクションを参照ください。

5.6 XMLドキュメントの入力ヘルパー

テキストビューとAuthentic ビュー内のXMLドキュメントでは、3つの入力ヘルパーを使用することができます: 要素入力ヘルパー、属性入力ヘルパー、そしてエンティティ入力ヘルパーです。要素入力ヘルパーにより要素が追加された場合、オプションダイアログの編集セクション1021で指定された設定に従い、必須の子要素や必須の属性、または全ての子要素を同時に自動的に追加する(または全く追加しない)ことができます。空の属性が追加された場合、引用句に追加されます。

他のビューでは入力ヘルパーの表示も、そのビューが持つ機能に合わせて変化することに注意してください。

要素入力ヘルパー

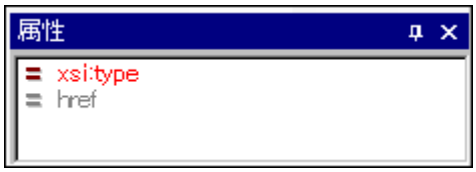
以下の点に注意してください:

- *テキストビュー:* カーソルの挿入ポイントに要素は挿入されます。使用されていない要素は赤色で、使用されているものは灰色で表示されます。必須の要素には要素名の隣に感嘆符(!)が表示されます。
- *Authentic ビュー:* 選択された要素の後、前、そして中に要素を挿入することができます。更に、ドキュメント内で現在選択されている要素がドキュメントのツリー構造にて表示されます。Authentic ビューにおける編集に関する詳細については、Authentic ビューセクションを参照ください。

属性入力ヘルパー

以下の点に注意してください:

- *テキストビュー:* カーソルが要素の開始タグ内にあるスペースの後にある場合、その要素に対して定義された属性が表示されます。使用されていない属性は赤色で、使用された属性はグレーで表示されます。必須の属性には属性名の隣に感嘆符(!)が表示されます。



  属性を挿入するには、必要とされる属性をダブルクリックしてください。カーソルがポイントされている個所で等号サインと属性の値を制限する引用符と共に属性が挿入されます。属性の値を直接入力できるようにカーソルを引用符の間をポイントしてください。
- *Authentic ビュー:* 要素が選択されているとき、その要素に対して宣言されている属性が表示されます。入力ヘルパーにて属性の値を入力してください。

エンティティ入力ヘルパー

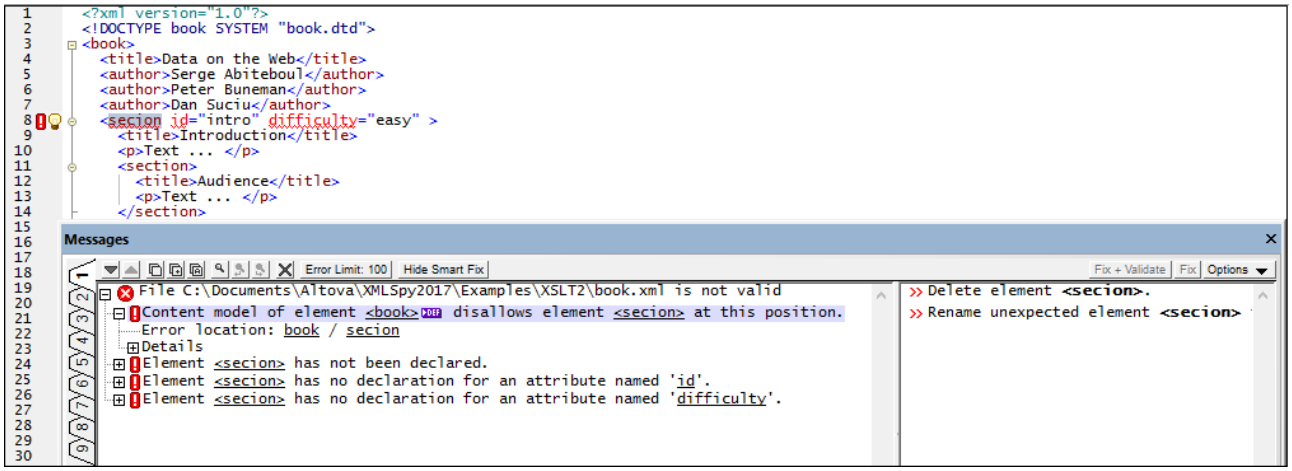
インライン(XMLドキュメント内)または外部DTD で宣言されたエンティティはエンティティ入力ヘルパーに表示されます。3つのビュー(テキスト、グリッド、Authentic)にて、入力ヘルパー内のエンティティをダブルクリックすることで、そのエンティティがカーソル位置に挿入されます。グリッドビューでは、エンティティは追加、挿入、子を追加タブに表示されます。

内部エンティティを追加した場合、そのドキュメントを保存して再度開かないと、エンティティはエンティティ入力ヘルパーに表示されません。

(C) 2015-2021 Altova GmbH

5.7　XML ドキュメントの検証

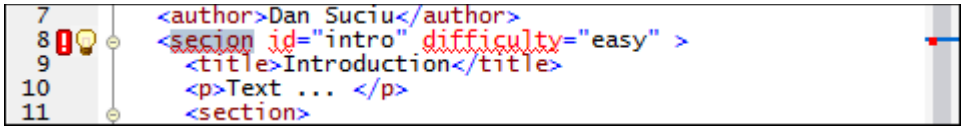
XML | 検証 (F8) コマンド は、ドキュメントを関連したDTD、XML スキーマ、または、他のスキーマに対して検証します。ドキュメントが妥当な場合、検証に成功したことを示すメッセージがメッセージウィンドウに表示されます。それ以外の場合、左側のペイン内にエラーの原因が表示されます (*下のスクリーンショットを参照してください*)。左側のペインで原因が選択されている場合、使用できる場合は、右側のペインに表示されているスマートフィックスを使用します。スマートフィックスの提案は、関連付けられたスキーマ内の情報をベースにしています。スマートフィックスを適用するには以下を行います (i) ダブルク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



エラーインジケーターとスマートフィックス

エラーインジケーター

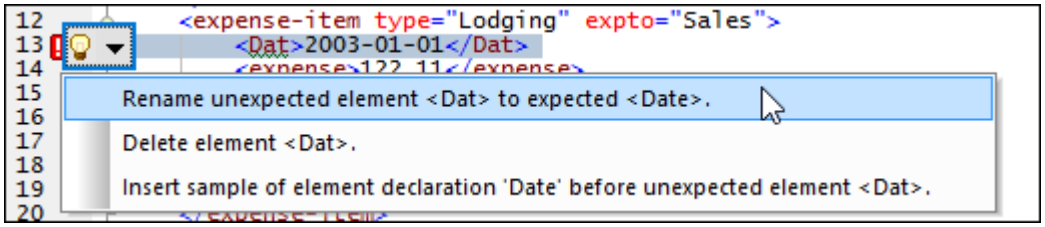
テキストビューでは検証エラーの追加インディケーターが2つ存在します (*下のスクリーンショット参照*): (I) ラインの付番余白内の赤い感嘆符 (ii) (ウィンドウの右側の) 赤色のマーカー正方形。



赤い四角はウィンドウの右側にあるスクロールバーの左横に表示されることに注意してください (*上のスクリーンショットを参照*)。スクロールバーは次の個別の2つの種類のマーカーを表示するためここで説明されています: (I) ハイライトされたテキストの発生個所 (茶色、スクロールバーの左側。*ドキュメントのナビゲート* 129)を参照してください。(ii) 発生個所を確認してください (茶色、スクロールバーの右側 *検索と置き換え* 132 を参照)

スマートフィックス

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



以下の点に注意してください：

- 検証エラーのインジケーターとクイックフィックスはXMLSpy 内で検証されるドキュメントの型、例えばJSON ドキュメントなど、に対して使用することができます。
- 上記の検証エラーインジケーターとスマートフィックスは、**XML | 検証** (F8) コマンドが実行された場合のみ更新されますが、背景では更新されません。ですから、エラーを修正した後、**検証** (F8) コマンドを再度実行し、エラーが修正されていることを確認してください。

XMLドキュメントの検証に関する詳細は、以下を参照してください：検証 841 コマンド。

編集中の検証

編集中に検証 モードはデフォルトでオンに切り替えられています。オンに切り替えられていると、テキストビューと（JSON グリッドビューの）ドキュメントの変更中に整形式チェックと検証チェックが実行されます。（整形式のチェックに加え）検証を実行するには、DTD またはXML スキーマがXMLドキュメントに割り当てられている必要があります（JSON スキーマはJSONドキュメントに割り当てられている必要があります）。エラーは左側の余白内のエラーを含むライン上でエラーインジケーターにより表示されています（上記参照）。

編集中に検証 モードは(i) 「XML | 編集中に検証」 847 メニューコマンド (ii) **編集中に検証** ツールバーボタン または(iii) オプションダイアログの検証設定 1015 の *編集時* オプションによりオンとオフに切り替えることができます。

5.8 XML フラグメントの挿入

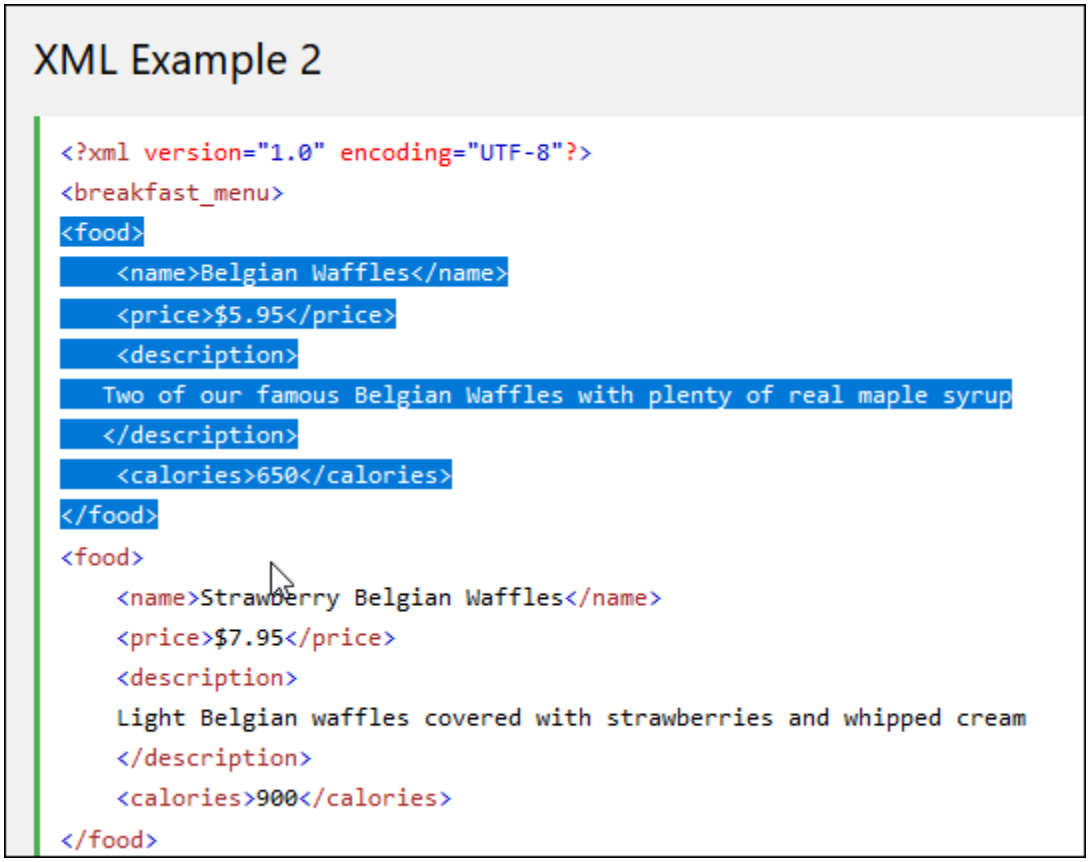
他のアプリケーションとWeb ページからXML フラグメントを挿入することができます。2つの方法を使用してこれらのフラグメントを挿入することができます:

- ドラッグアンドドロップを使用してテキストビューとグリッドビューに挿入する。グリッドビューにドラッグアンドドロップする場合、ドラッグオーバーレイ内で使用可能なインテリジェントな情報 246 がフラグメントをドロップする箇所を決定する手助けをします。
- コピーして貼り付けを使用してテキストビューとグリッドビューに挿入する。

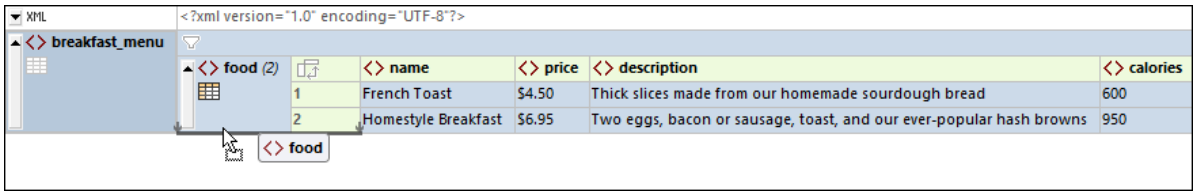
サンプル

次のサンプルはフラグメントをXMLドキュメント内の正確な箇所に素早く追加する方法を示しています。

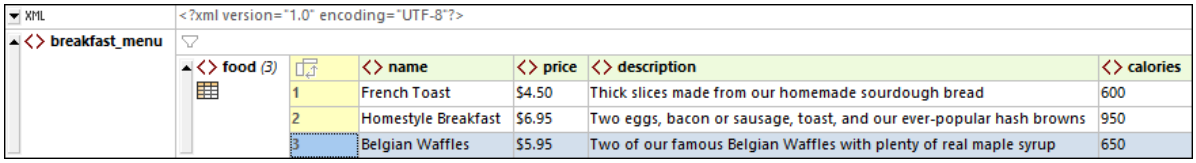
1. 下でハイライトされているフラグメントが(w3schools.com でのXML チュートリアルから) 選択されています。子要素の数値を含む food という名前の要素です。



2. 下のスクリーンショットは2個の food 要素がテーブルの行として表示されているXMLドキュメントのグリッドビューを示しています。Web ページからXML フラグメントが food テーブルにドラッグされる際、ドラッグされたフラグメントが food 要素として food テーブルに最後の行として挿入される情報を含むドラッグオーバーレイ 485 が表示されます。



3. *フラグメントがドロップされると、* **`breakfast_menu`** *の* **`food`** *子要素として希望する箇所に配置されます(下のスクリーンショット参照)。*



(C) 2015-2021 Altova GmbH

5.9 XSLT と XQuery による処理

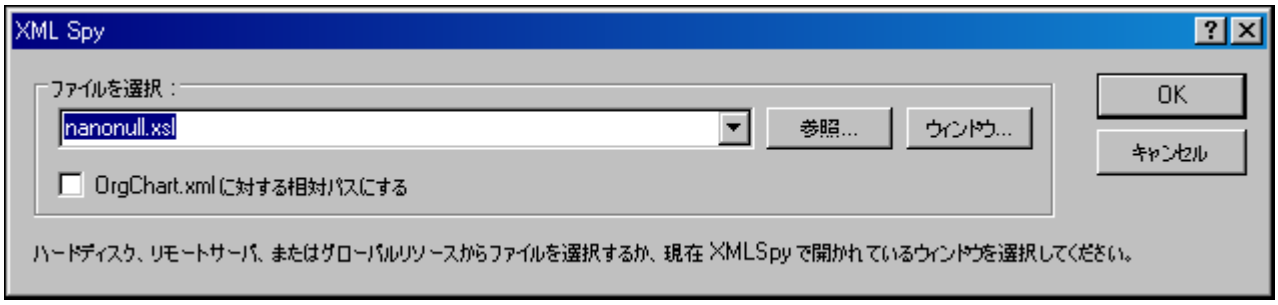
XML ドキュメントと XSLT または XQuery ドキュメントを組み合わせることで、出力ドキュメントを生成することができます。XMLSpy には XSLT 1.0、XSLT 2.0、そして XQuery 1.0 プロセッサーが内蔵されており、GUI から以下のような機能を利用することができます:

- XSLT スタイルシートの割り当て 275
- XSLTへ移動 275
- XSLT パラメーターとXQuery 変数 275
- XSLT 変換 276
- XQuery 実行 276
- RaptorXMLを使ったXML 処理の自動化 276

XSLT スタイルシートの割り当て

「XSLT/XQuery | XSL の割り当て」コマンドを使用(以下のスクリーンショットにあるように、ダイアログにてファイルを指定)して、XSLT スタイルシートを XML ドキュメントに割り当てることができます。割り当ては W 3C により定義された `xml-stylesheet` 処理命令 (PI) として XML ドキュメント内に記述されます。
XSLT 変換が呼び出された際(「XSL/XQuery | XSL 変換」)に、この割り当ては使用されます。

更に、「XSL/XQuery | XSL:FO 変換」コマンドにより(以下のスクリーンショットにあるようにダイアログにてファイルを指定して) XSLT-FO スタイルシートを割り当てることもできます。割り当ては Altova により定義された altova_ xslfo 処理命令 (PI) として XML ドキュメント内に記述されます。
XSLT-FO 変換が呼び出された際(「XSL/XQuery | XS:FO 変換」)に、この割り当ては使用されます。



XSLT ファイルのソースとしてグローバルリソースを選択することもできます。グローバルリソースはファイルまたはフォルダーに対するエイリアスです。アクティブなグローバルリソースの構成をマウス操作(メニューコマンド「ツール | アクティブな構成」)により変更することで、ターゲットファイルまたはフォルダーを変更することができます。グローバルリソースを使うことで、割り当てられた XSLT ファイルを切り替え、試験などに利用することができます。詳しいグローバルリソースの使用方法に関しては、Altova グローバルリソース 660 の項を参照ください。

以前の割り当てにより作成された上記の処理命令が存在する場合、それら既存の割り当てを上書きしても良いか確認されます。

XSLT へ移動

「XSL/XQuery | XSL へ移動」コマンドにより、XML ドキュメントに割り当てられた XSLT ファイルを開くことができます。

XSLT パラメーターとXQuery 変数

「XSL/XQuery | XSLT パラメーター/XQuery 変数」コマンドを選択することで表示されるダイアログにより、XSLT パラメーターならびに XQuery 変数の定義、編集、そして削除を行うことができます。ここで定義されたパラメーター、変数の値は、XMLSpy における全ての XSLT 変換ならびに XQuery 実行により使用されます。しかし、これらの値は MSXML の様な外部エンジンには受渡されません。この機能の詳細については、ユーザーリファレンス 890 の項を参照ください。

XSLT 変換

2種類のXSLT 変換を利用することができます:

- 標準 XSLT 変換(「XSL/XQuery | XSL 変換」):変換の出力は新規ウィンドウに表示されるか、スタイルシートで指定されている場合、指定された場所にファイルが保存されます。変換のために使用されるエンジンはオプションダイアログ(「ツール | オプション」[1014])のXSL タブ[1036]にて指定することができます。
- XSL-FO 変換(「XSL/XQuery | XSL:FO 変換」):XML ドキュメントは2段階の処理によりPDF へと変換されます。最初のステップでは、オプションダイアログ(「ツール | オプション」[1014])のXSL タブ[1036]で指定されたXSLT プロセッサーによりXML ファイルがFO ドキュメントに変換されます。オプションタブの下部にてFOP の様なFO プロセッサーに同封されたXSLT エンジンも使用することができます。2番目のステップでは、オプションダイアログ(「ツール | オプション」[1014])のXSL タブ[1036]で指定されたFO プロセッサーによりFO ドキュメントからPDF 出力が生成されます。

注意: XSL: FO 変換コマンドにより、(XML ドキュメントの一種である)FO ドキュメントをPDF に変換することができます。変換のソースドキュメントがFO ドキュメントである場合、上で紹介された2 段階処理の2 番目のステップが直接実行されます。

XQuery 実行

「XSL/XQuery | XQuery 実行」コマンドをクリックすることで、アクティブな XML ドキュメント上でXQuery ドキュメントを実行することができます。XQuery ファイルを入力するよう促され、実行結果のドキュメントは新しいウィンドウに表示されます。

RaptorXML を使ったXML 処理の自動化

Altova RaptorXML はXML 検証、XSLT 変換、およびXQuery 実行を搭載したアプリケーションです。コマンドライン、COM インターフェース、Java プログラム、そして.NET アプリケーションから使用することができ、XSLT 変換タスクはRaptorXML により自動化することができます。例えば、RaptorXML によりドキュメントのセットを変換するバッチファイルを作成することができます。詳細については RaptorXML ドキュメンテーション を参照ください。

5.10 PDF フォント

フォーマッターとPDF ビューアーに対するフォントの使用
(例えばFOP といった)フォーマッターによりPDF が作成され、PDF ビューアー(例えばAdobe Reader)にて確認することができます。

PDF のレイアウトを行うには、ドキュメント内で使用されているフォント、その中でも特に、使用されている文字の幅に関する詳細をフォーマッターへ伝える必要があります。この情報を使用することで行の長さやハイフンの位置、行端揃えなどの計算が行われます。この情報はフォントメトリックと呼ばれ、各フォントとともに保管されます。フォーマッターによっては、システムのフォントフォルダーからメトリックを直接読み込むことができますが、(FOP のような)それ以外のフォーマッターに対しては、定められた形式でメトリックを渡す必要があります。フォントのメトリックがフォーマッターに対して与えられると、期待されるかたちでPDF のレイアウトが作成されることになります。そのため、ドキュメントにて使用されている全てのフォントに対するメトリックファイルが、使用しているフォーマッターに対して渡されるようにする必要があります。

フォーマッターはフォントを参照することができるほか、PDF ファイル内部にフォントを埋め込むことができます。フォントが参照された場合、PDF ビューアー(例: Adobe Reader)は通常(base 14 フォントが含まれる)独自のリソースフォルダーからそのフォントの検索を行い、フォントがそこに無い場合はシステムのフォントフォルダーを検索します。フォントが利用できる場合、PDF が表示される際にそのフォントが使用されます。フォントが見つからない場合、リソースフォルダー上にある代替フォントが使用されるか、エラーが表示されます。代替フォントには異なるメトリックが指定されている場合もあり、表示エラーが生成される可能性もあります。

フォーマッターによりフォントがPDF ファイル内部に埋め込まれた場合、埋め込まれたフォントがPDF ビューアーにより使用されます。フォーマッターはフォントの全文字セットを埋め込むことができるほか、ドキュメント内部で使用されているフォントのサブセットだけを埋め込むことができます。フォントの埋め込み方法により、PDF ファイルの大きさが変わってくるだけではなく、フォントの使用に関する版権にも影響を与える可能性があります(以下のメモを参照ください)。フォーマッターのオプションを指定する際に、これらオプションの選択を指定できる場合があります。

XMLSpy とPDF フォント

XMLSpy では、XSL-FO ドキュメント(以下 FO ドキュメント)からPDF が生成されます。FOP といった外部 FO プロセッサーにより、XSL-FO ドキュメントが処理されます。オプションダイアログでは、FO プロセッサーの場所を指定することができます。これによりXMLSpy の GUI からFO の処理を開始することができます。

XSL-FO ドキュメントそのものはXML ドキュメントをXSLT スタイルシートで処理することにより生成されます。(XMLSpy に搭載されている)Altova のXSLT エンジンを使用することができるほか、外部 XSLT エンジンを使用することができます。

全テキストのフォントプロパティを含むPDF ドキュメントのフォーマットは、XSL-FO ドキュメントにて指定されます。使用しているフォーマッターがフォントのメトリックをフォントから直接読み取ることができる場合、フォントにアクセスするためにフォーマッターのセットアップを行う必要があります。FOP をフォーマッターとして使用している場合、base-14 フォントに含まれないフォントに対して、適切なフォントメトリックフィルを指定する必要があります。

フォーマッターがフォントを利用できるようにする

(FOP を含む)ほとんどのフォーマッターでは既にbase 14 フォントを利用することができ、これらのフォントをフォーマッターへ指定するために使用される名前を知ることが重要になります。base 14 フォントはフォーマッターによりサポートされる基本的なフォントですが、フォーマッター固有の方法に従うかたちで、これらのフォント以外にもフォーマッターが扱うことのできるフォントの数を増やすことができます。以下ではFOP にてフォントの数を増やす方法について記述されます。

FOP に対して新たなフォントサポートを追加するためのセットアップ手順

以下のステップにより、FOP が使用することのできるフォントを追加することができます:

1. PostScript またはTrueType フォントファイルから、フォントのメトリックファイルを生成します。FOP に搭載されているPFM Reader ならびにTTF Reader ユーティリティにより、PostScript ならびにTrueType フォントをXML フォントメトリックファイルへ変換することができます。詳細については、FOP: Fonts ページを参照ください。
2. FOP 構成ファイルをセットアップすることで、目的のフォントメトリックファイルを使用することができるようになります。FOP 構成ファイルにてフォントファイルに関する情報を入力することになります。FOP: Fonts ページを参照ください。
3. `fop.bat` ファイルにて、以下にある最後の行を:

```
"%JAVACMD%" […] org.apache.fop.cli.Main %FOP_CMD_LINE_ARGS%
```

構成ファイルの場所を含むように変更します:

```
"%JAVACMD%" […] org.apache.fop.cli.Main %FOP_CMD_LINE_ARGS% -c conf\fop.xconf
```

メトリックファイルがFOP(FOP 構成ファイル)にて登録され、FOP 実行ファイルが構成ファイルを読み取るようにセットされることで、指定されたフォントをPDF の作成に使用することができるようになります。

FOP 構成ファイルのセットアップ

FOP 構成ファイルは fop.xconf という名前で、FOP インストールフォルダー内にある conf フォルダー以下に配置されています。XML 形式により記述されるこのファイルを編集することで、FOP によりフォントメトリックファイルが適切に読み取られるように指定する必要があります。ドキュメント内部にある font 要素プレースホルダにて示される場所に、レンダリングを行う各要素に対応した font 要素を追加する必要があります:

```
<font metrics-url="arial.xml" kerning="yes" embed-url="arial.ttf">
          <font-triplet name="Arial" style="normal" weight="normal"/>
          <font-triplet name="ArialMT" style="normal" weight="normal"/>
</font>
```

上の例は、以下のように理解することができます:

arial.xml	メトリックファイルのURL です。絶対パスの使用が推奨されます。
arial.ttf	TTF ファイルの名前です(通常 %WINDIR%\Fonts 以下に収められています)。
Arial	フォントファミリーに Arial が指定されている場合、上記のメトリックと TTF ファイルが使用されるように指定します。
style="normal"	フォントスタイルが(italic などではなく) normal として定義されている場合、上記のメトリックと TTF ファイルが使用されるように指定します。
weight="normal"	フォントウェイトが(bold などではなく) normal として定義されている場合、上記のメトリックと TTF ファイルが使用されるように指定します。

フォントの版権に関するメモ: フォントの使用は著作権法の対象となり、使用に関する条件は一定ではありません。フォントの埋め込みを行う場合、特にフォント全体の埋め込みを行う場合、購入したフォントをそのように使用しても問題が無いか確認する必要があります。

文字セット

フォントの文字セットも考慮する必要がある点に注意してください。base 14 フォントでは、ISO-8859-1 により規定される文字の他に、シンボルとZapf Dingbats フォントにある文字がカバーされます。base 14 フォントによりカバーされない文字がドキュメントに含まれる場合、その文字が含まれる文字セットを含んでいるフォントを使用する必要があります。Arial Unicode のようなフォントには、Unicode によりカバーされる文字が含まれます。

5.11 その他の機能

今までご紹介した以外にも、XML ファイルを使用するうえで以下の機能を利用することができます。

- エンコーディング 279
- DTD やスキーマの生成 279
- 検索と置換 279
- XPath の評価 280
- テキストのインポートとエクスポート 280

エンコーディング

XML ファイル(そしてその他のドキュメント)のエンコーディングはメニューコマンド「ファイル | エンコード」 751 により設定することができます。XML ならびに非 XML ファイルのデフォルトエンコーディングは「オプション | エンコード」 1020 セクションにより指定することができます。

DTD やXML スキーマの生成

「DTD/スキーマ | DTD/スキーマの生成」 854 メニューコマンドにより、XML ドキュメントの構造を記述するスキーマを作成することができます。表示される DTD スキーマの生成ダイアログでは、生成するスキーマの種類(DTD もしくは XML スキーマ)や、XML ドキュメントにある値から列挙を生成するなどの XML スキーマオプションを指定することができます。

検索と置換

「編集」メニューからアクセスすることができる検索 773 ならびに置き換え 776 機能によりパワフルな検索機能を手に入れることができます。検索の対象となる単語には、大文字や小文字の区別をする、単語の完全マッチ、更には正規表現を使用することもできます。検索の対象はドキュメント内の選択や(以下のスクリーンショットに示されるように)特定のノード型に限定することができます。

上のスクリーンショットはグリッドビュー内に検索ダイアログを表示しています。。ダイアログとその機能は、アクティブになっているビューにより変化します。テキストビューの検索機能については、ここ132を参照してください。

XPath の評価

XPath ウィンドウで入力した XPath 条件式をアクティブな XML ドキュメントに対して評価することができます。評価結果は XPath ウィンドウ内に表示され、その中のノードをクリックすることで、メインウィンドウ内の対応するノードがハイライトされます。XPath ウィンドウは「XML | XPath を評価」コマンドによりアクティブにすることができます。

テキストのインポートとエクスポート

テキストデータは他のアプリケーションから(そして他のアプリケーションへ)インポート(ならびにエクスポート)することができます。これらの機能のコマンドは変換938メニューから利用することができます。

6 DTD と XML スキーマ

Altova Web サイト: XML スキーマエディター

このセクションでは DTD 282 ならびに XML スキーマ 284 に関する概要を説明します。SchemaAgent 289 ならびにパワフルなスキーマの内を検索 307 する機能についても解説されます。編集機能の他に、XMLSpy ではDTD/スキーマに関する以下の機能が提供されます:

カタログメカニズム

OASIS カタログ機能 301 により URI からローカルアドレスのリダイレクトが可能になり、複数のワークステーションにおける利用が促進されます。

スキーマサブセット

スキーマビューでは、巨大なスキーマのコンポーネントを、別のファイルとして作成することができます。これら(より小さな)スキーマサブセットを、大きなスキーマにインクルードすることもできます。また、スキーマのフラット化という名前の機能を使うことで、逆の操作を行い、インクルードされているファイル内にあるコンポーネントを、直接巨大なスキーマに取り込むことができます。スキーマのサブセットと、スキーマのフラット化に関する詳細な情報は、スキーマサブセット 285 のセクションを参照ください。

DTD とXML スキーマ間の変換

「DTD/スキーマ 852 」メニューを使用することで、DTD とXML スキーマ間の変換を行うことができます。使用方法などDTD がフラット化されている場合、含まれているインポートされたモジュールは親ファイルに直接保存され、使用されていないコンポーネントは削除されます。

サンプルXML ファイルの生成

「DTD/スキーマ | サンプルXML ファイルの生成」 860 メニューコマンドにより、アクティブな DTD やXML スキーマファイルをベースにしたスケルトンXML ドキュメントの生成を行うことができます。スキーマをベースにしたXML ファイルを作成するのにとても便利な機能です。

定義へ移動

カーソルがXML ドキュメントのノード内に位置している場合、「DTD/スキーマ | 定義へ移動 854 」メニューコマンドにより、スキーマファイルが開かれ、選択されているXML ノードの定義を行っている箇所がハイライトされます。

6.1 DTD

テキストビューとグリッドビューでDTD ドキュメントを編集することが可能です。デフォルトのビューはオプションダイアログのファイル型セクション[1018] で設定することができます。

テキストビュー

テキストビューでは、ドキュメントは構文により色分けされています。以下はDTD の断片サンプルです。:

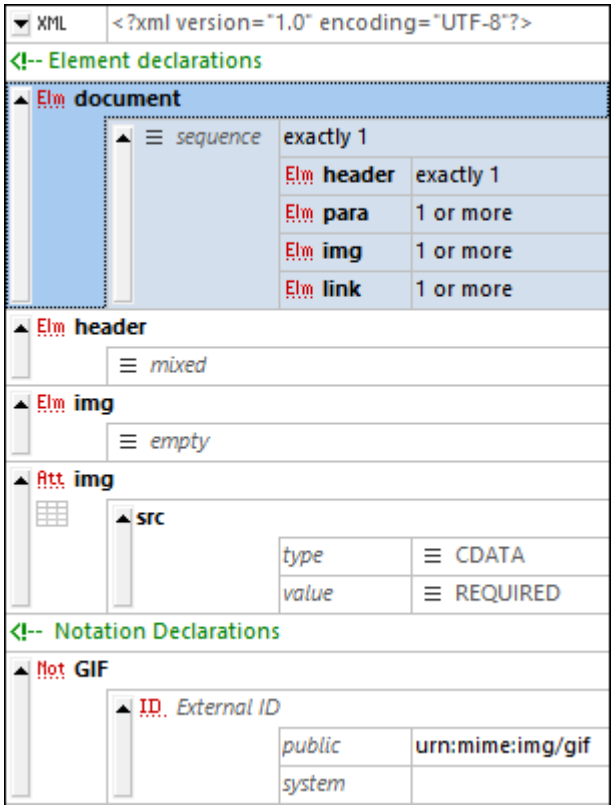
```
<?xml version="1.0" encoding="UTF-8"?>
<!--Element declarations-->
<!ELEMENT document (header, para+, img+, link+)>
<!ELEMENT header (#PCDATA)>
<!ELEMENT img EMPTY>
  <!ATTLIST img
    src CDATA #REQUIRED
  >

<!-- Notation Declarations -->
<!NOTATION GIF PUBLIC "urn:mime:img/gif">
```

インデントはインデントガイドにより表され、タブ キーにより入力することができます。インデントの大きさはテキストビュー設定ダイアログ[124] にてセットすることができます。

グリッドビュー

グリッドビューにおいて、DTD ドキュメントはテーブルとして表示されます。以下のスクリーンショットでは、上にあるDTD をグリッドビューで表示したものです。



DTD 構造の編集

- コンテキストメニューまたは **XML** メニューを使用してカーソルがセル内にある場合、ノードを挿入または追加、または子ノードを追加することができます。
- セルの左上のノードの型アイコンをクリックしてノードの型を変更します。
- (*シーケンス*、*複合*、*空* などの) コンテンツモデルとノードの発生修飾子 (*厳密に 1*、*1 またはそれ以上* など) を対応するアイコンとクリックして、オプションを選択して変更することができます。
- ドラッグアンドドロップを使用してノードをドキュメント内の新規の場所に移動、および新規の場所にコピーして貼り付けることができます。

DTD 値の編集

- 要素、属性名、およびコメントなどの値を編集する場合セルをダブルクリックして編集します。

グリッドビューツールバー
グリッドビューツールバーではビューの設定ダイアログへのアクセスを提供され、また、倍率の設定やワードラップなどのコマンドが含まれています。

XMLSpy 内の DTD 機能

XMLSpy には以下の非常に便利な機能が含まれています::

- *DTD を XML スキーマへ変換:* 「DTD/スキーマ | DTD/スキーマの変換」 856 コマンドにより、DTD を XML スキーマへ変換することができます。
- *DTD からサンプル XML ファイルを生成*「DTD/スキーマ | サンプル XML ファイルの生成」 860 コマンド により、アクティブな DTD をベースにした XML ドキュメントを生成することができます。

6.2 XML スキーマ

XML スキーマドキュメントの編集はテキストビュー、グリッドビュー、そしてスキーマビューで行うことができます。XML スキーマドキュメントが開かれるデフォルトのビューは、オプションダイアログのファイルの型タブ[1018]にて設定することができます。ドキュメントの編集中でも、その時に一番使いやすいビューに切り替えて作業を行うことができます。XML スキーマドキュメントは、大抵の場合 `.xsd` または `.xs` のファイル拡張子で保存されます。

テキストビューでの編集

テキストビューの XML スキーマ編集は XML ドキュメントと同じように行われ、XML ドキュメントで使用可能な編集機能[234]を XML スキーマでも利用することができます。XML ドキュメントと同様に、参照しているスキーマを特定し、アクセスすることができる場合、カーソルの位置に追加することができるアイテムがテキストビュー入力ヘルパー[137]に表示されます。

グリッドビューでの編集

グリッドビューの XML スキーマ編集は XML ドキュメントと同じように行われ、XML ドキュメントで使用可能な編集機能[141]を XML スキーマでも利用することができます。グリッドビューにてアイテムが選択されている場合、カーソルの位置に追加することのできるアイテムがグリッドビュー入力ヘルパー[147]に表示されます。

スキーマビューでの編集

スキーマビューはスキーマのデザインをグラフィカルに行うためのビューです。スキーマビューにてスキーマの作成、編集を行うと、対応するテキストドキュメントが XMLSpy により生成されます。スキーマビューにて XML スキーマドキュメントを作成、編集する詳細な方法については編集ビュー | スキーマビュー[151]を参照ください。

XMLSpy での XML スキーマ機能

XMLSpy には更に、以下のような便利な機能が含まれます:

- *XML スキーマを DTD へ変換*:「DTD/スキーマ | DTD/スキーマの変換」[858] コマンドにより、XML スキーマを DTD へ変換することができます。
- *スキーマからサンプル XML ファイルを生成*:「DTD/スキーマ | サンプル XML ファイルの生成[860]」コマンドにより、アクティブな XML スキーマをベースにした XML ドキュメントを生成することができます。生成されるサンプル XML の要素ならびに属性のサンプル値も指定することができます。

(C) 2015-2021 Altova GmbH

6.3 スキーマサブセット

XML スキーマのコンポーネントは、スキーマサブセットと呼ばれる、(より小さな)外部スキーマファイルとして作成することができます。小さなスキーマサブセットの集合を(インクルードと一緒に)使用することで、巨大なスキーマを使用した際に比べて、単一の巨大なスキーマよりもスキーマの管理を行ないやすくなります。

スキーマビューにて以下のような操作を行うことで、スキーマサブセット機能が持つ機能を理解することができるはずです:

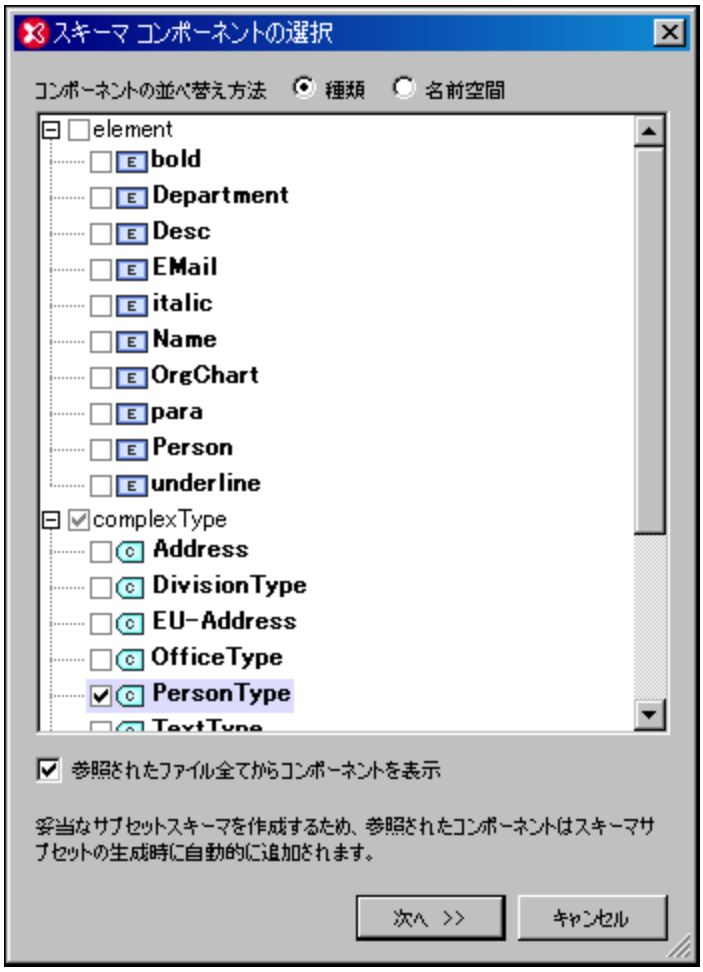
1. アクティブなスキーマのコンポーネントを含むスキーマサブセットを作成します。具体的な操作の方法については以下を参照ください 285。
2. 必要に応じて別のスキーマサブセットを作成します。
3. 新たに作成されたスキーマサブセットをインクルードして、大きなスキーマを形成します。スキーマ概要ウィンドウ 156にて、インクルードコンポーネントを追加または挿入し、新たに作成したスキーマサブセットファイルを選択する操作を、各スキーマサブセットに対して行ないます。
4. オリジナルの巨大なスキーマに元々存在し、インクルードされたサブセットにも存在するコンポーネントを削除します。

スキーマビューでは、上の操作とは逆に、インクルードされたファイルをフラット化することができます。つまり、(i) スキーマサブセットに含まれているコンポーネントがメインスキーマに直接追加し、(ii) インクルードされたスキーマサブセットをメインスキーマから削除します。スキーマをフラット化する方法については以下に記述されます 287。

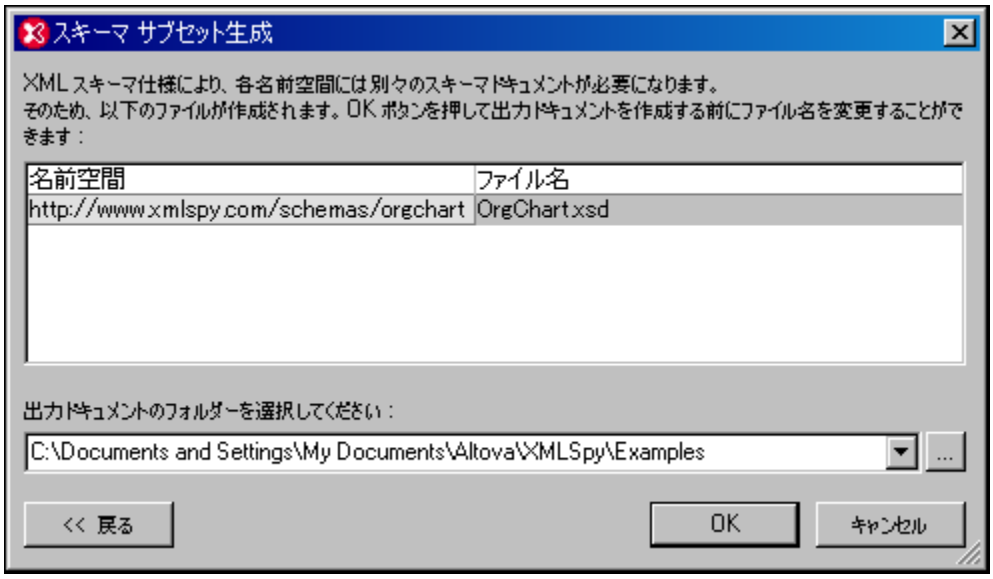
スキーマサブセットの作成

スキーマサブセットを作成するには:

1. 目的のXML スキーマがスキーマビューにてアクティブな状態で、メニューコマンドの「**スキーマ設計 | スキーマサブセットの作成**」を選択します。この操作により、スキーマコンポーネントの選択ダイアログが表示されます(以下のスクリーンショットを参照)。
2. ダイアログにて、単一のスキーマサブセットとして作成したいコンポーネントにチェックを入れ、「**次へ**」をクリックします。ペイン以下のチェックボックスにより、参照されている全てのファイルにあるコンポーネントを表示することができます。



3. （以下のスクリーンショットにある）スキーマサブセット生成ダイアログにて、スキーマサブセット パッケージの名前を入力します。同時に、新たなスキーマサブセット ファイルが保存されるフォルダーも指定する必要があります。オリジナルのスキーマにてインポートされていたコンポーネントが、作成されるコンポーネントに含まれている場合、スキーマサブセット パッケージには複数のファイルが含まれることになります。スキーマサブセットでは、各名前空間に対して、異なるファイルが作成されます。デフォルトでダイアログに表示されるファイル名はオリジナルのファイル名となりますが、オリジナルのファイルを上書きすることは許されないため、オリジナルのファイルがあるフォルダーにサブセット ファイルを保存したい場合は、名前の変更を行う必要があります。

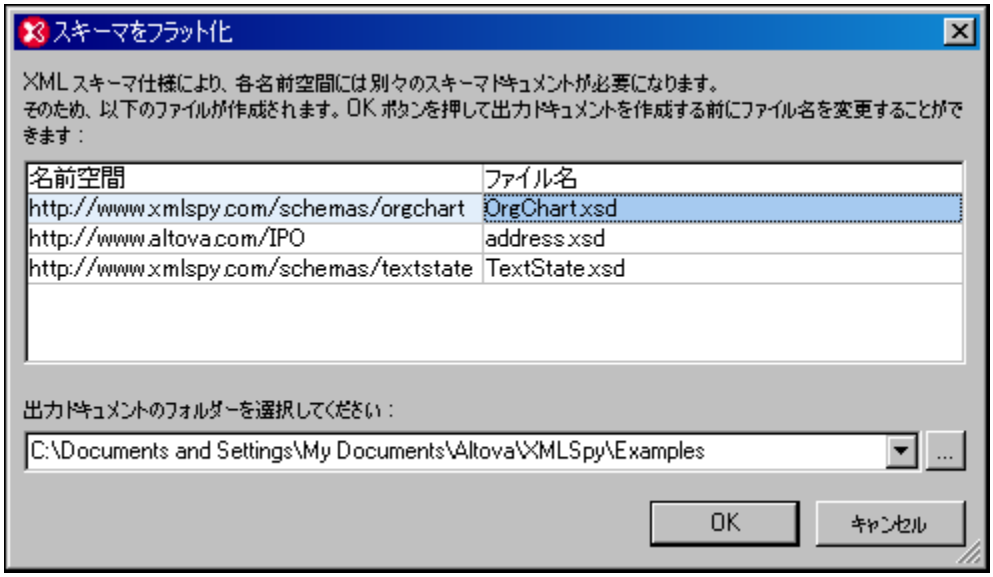


4. 「OK」をクリックすることで、アクティブなファイルの名前空間を持つスキーマサブセットファイルが、スキーマビューで開かれます。パッケージ内にあるその他のファイルは、作成されますが、スキーマビューでは開かれません。

スキーマのフラット化

スキーマビューにてアクティブなスキーマをフラット化すると、(i) 各スキーマのコンポーネントがアクティブなスキーマのグローバルコンポーネントとして追加され、(ii) インクルードされたスキーマが削除されます。

アクティブなスキーマをフラット化するには、メニューコマンドから「**スキーマ設計 | スキーマをフラット化**」を選択します。この操作により、(以下のスクリーンショットにある) スキーマをフラット化ダイアログが表示され、各ファイルの名前が、フラット化されたスキーマで使用される名前空間と共に表示されます。デフォルトのファイル名はオリジナルのファイル名のままになっていますが、オリジナルのファイルの場所と同じフォルダーに保存しようとする場合には、ファイル名を変更する必要があります。フラット化されたスキーマと、関連するファイルを保存するフォルダーを指定することもできます。



「OK」をクリックすると、フラット化されたスキーマが、スキーマビューで開かれます。

6.4 SchemaAgent との連携

XMLSpy は Altova の SchemaAgent テクノロジーと連動するように設定することができます。

SchemaAgent テクノロジー

SchemaAgent テクノロジーにより、複数のスキーマ間のリレーションシップを構築することができます。以下により構成されています:

- (フォルダーやネットワークの)検索パス上にある、複数のスキーマ間のリレーションシップに関する情報を保持、提供する SchemaAgent Server。
- A SchemaAgent クライアント、(i) スキーマ間のリレーションシップを作成し、(ii) これらのスキーマを管理するために接続されている (スキーマの名前の変更、移動、削除する)SchemaAgentサーバーからの情報を使用する Altova の SchemaAgent 製品。

SchemaAgent サーバーの2 つの種類を使用することができます:

- ネットワーク上にインストールし、利用することのできる Altova SchemaAgent Server。
- SchemaAgent クライアント製品である Altova SchemaAgent 。より軽量なサーバーである LocalServer が搭載されており、SchemaAgent がインストールされているマシンからのみアクセスすることができます。

XMLSpy から SchemaAgent テクノロジーを利用することで、SchemaAgent サーバーから得られるスキーマ情報を使用して、スキーマをスキーマビューにて直接編集することができます。ここでは、XMLSpy が SchemaAgent サーバーに接続し、SchemaAgent Client とのやり取りを行いながら SchemaAgent Server へリクエストを送信します。XMLSpy が SchemaAgent と連動するよう設定された場合、スキーマビューの入力ヘルパーには、現在のスキーマビューでアクティブなスキーマから得られたコンポーネントだけでなく、接続している SchemaAgent サーバーの検索パスにあるその他のスキーマから得られたコンポーネントも表示されます。この機能を使うことで、これらのコンポーネントに直接アクセスすることができます。他のスキーマに属しているコンポーネントのコンテンツモデルを、スキーマビューで確認し、これらコンポーネントを再利用することができます。スキーマ間のリレーションシップを構築して、XMLSpy から複雑なスキーマの管理を直接行うこともできます。

SchemaAgent と SchemaAgent Server のインストール

SchemaAgent ならびに SchemaAgent Server のインストール、そしてサーバー上の検索パスの設定については、SchemaAgent ユーザーマニュアルを参照ください。

XMLSpy を SchemaAgent クライアントとしてセットアップする

XMLSpy が SchemaAgent クライアントと連動して使用するには以下の操作を行う必要があります:

- Altova ウェブサイトから SchemaAgent をダウンロードします。SchemaAgent の LocalServer を使用することでスキーマを提供することができるようになります。LocalServer にて検索パスを設定する方法については SchemaAgent のユーザーマニュアルを参照ください。LocalServer 上の検索パスの構成に関する詳細は SchemaAgent ユーザーマニュアルを確認してください。
  注意: 無料トライアル期間が終了した後に SchemaAgent を使用するには有効なライセンスを購入する必要があります。Altova MissionKit 製品パッケージ Enterprise Edition には SchemaAgent 製品とライセンスキーが含まれています。(SchemaAgent Server アプリケーションは Altova MissionKit パッケージには含まれていません。)
- 上記に加えて、必要ならば Altova ウェブサイトからネットワークベースの SchemaAgent Server をダウンロードしてください。
- SchemaAgent(そして SchemaAgent Server) の検索パスを定義します。検索パスの詳細な定義方法については SchemaAgent ユーザーマニュアルを参照ください。(検索パスとはリレーションシップのマッピングに使われる、スキーマが含まれるフォルダーへのパスのことです)。
- XMLSpy 内から SchemaAgent への接続を開始します。

重要点: Altova から提供される SchemaAgent ならびに SchemaAgent に関連する製品(XMLSpy を含む)のバージョン 2005 リリース 3 以降の製品は、それ以前のバージョンの SchemaAgent または SchemaAgent に関連する製品との**互換性を持ちません**。

XMLSpy 内の SchemaAgent コマンド

XMLSpy 内の SchemaAgent 機能はスキーマビューからのみ利用することができ、スキーマ設計メニューのメニューコマンド（*スクリーンショットを参照*）ならびにスキーマビューの入力ヘルパーからアクセスすることができます。

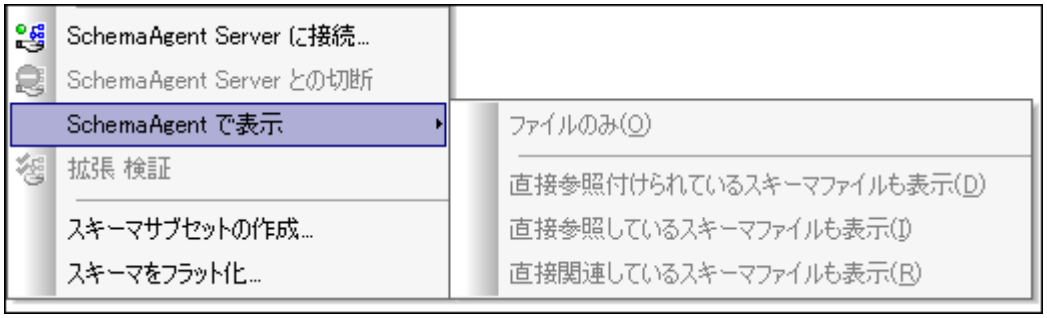
メニューコマンドでは一般的な管理機能が提供されます。入力ヘルパー（ならびにドラッグアンドドロップといった GUI を使った機能）によりスキーマの編集を実際に行うことができます。

このセクションではスキーマビューで利用可能な SchemaAgent の機能の使用方法について説明します。

6.4.1 SchemaAgent Server へ接続

メモ 接続を確立するには SchemaAgent Client がインストールされている必要があります。

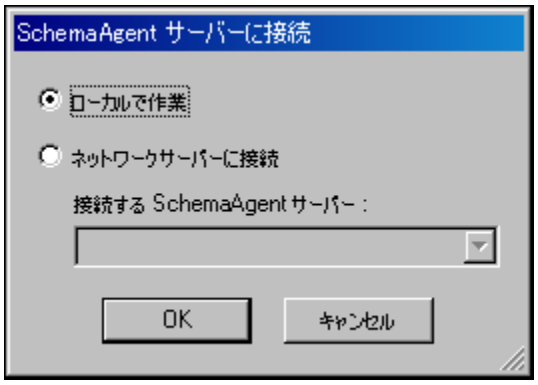
SchemaAgent Server に接続する前に only the **SchemaAgent Server に接続** コマンドのみが **スキーマ設計** メニュー内で使用することができます。**スキーマ設計** メニュー内の他の SchemaAgent コマンドは無効化されています（*スクリーンショット参照*）。SchemaAgent Server への接続が確立されれば、その他のメニューアイテムも選択可能になります。



接続方法

SchemaAgent サーバーへ接続するには、以下の操作を行ってください:

1. Click the Connect to SchemaAgent サーバーに接続アイコン (「**スキーマ設計 | SchemaAgent Serverに接続**」)をクリックします SchemaAgent Server への接続ダイアログが開かれます（*下のスクリーンショット*）:

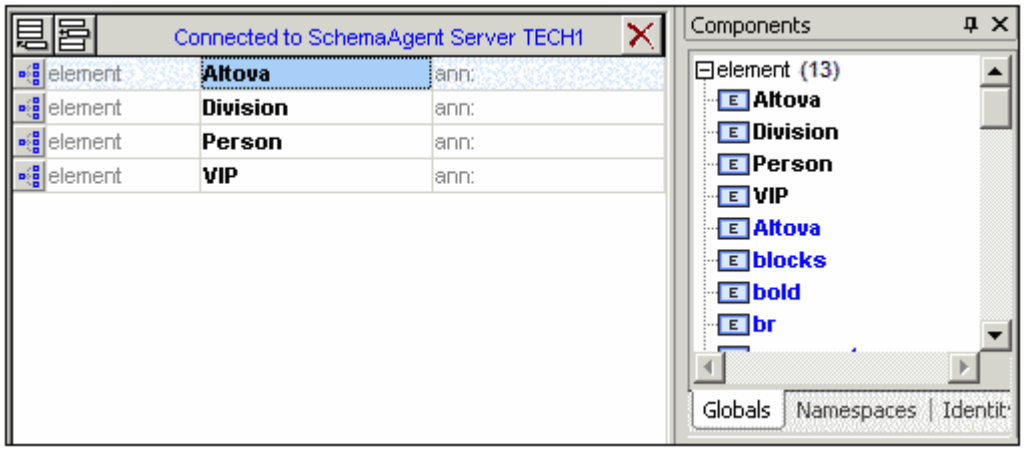


2. ローカルサーバー(Altova SchemaAgent に同封されている SchemaAgent サーバー)またはネットワークサーバー(無料で利用することのできる Altova SchemaAgent Server 製品)を使用することができます。ローカルで作業を選択し、**OK**を押すと、SchemaAgent のローカルサーバーが起動され 接続が確立されます。ネットワークサーバーへ接続を選択する場合、選択された SchemaAgent Server が起動していなければなりません。

> **Windows XP SP2 上で動作しているサーバーについて**
> SchemaAgent Server 名が SchemaAgent Server への接続ダイアログにリストされていても接続できない場合、サーバーがネットワークの名前解決プロセスの一部ではない場合があります。Windows XP SP2 ではファイヤーウォールにより名前解決をデフォルトでブロックしています。
>
> このようなサーバーに接続するには、以下のうちどれかを行ってください:
>
> - サーバーの設定を変更して、名前解決を有効にする
> - サーバーの IP アドレスを 接続ダイアログボックスの編集フィールドに入力します。
>
> SchemaAgent Client には最後に成功した接続の接続文字列が保存されるため、この操作は1度だけ行う必要があります。

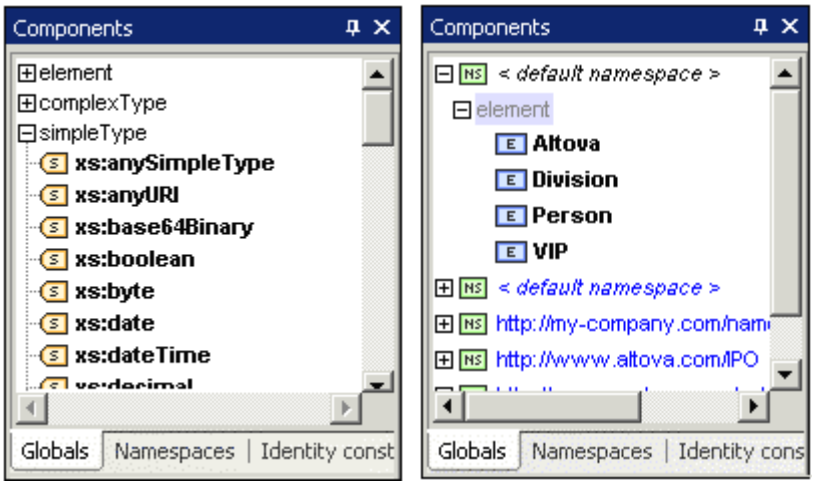
SchemaSchemaAgent サーバーに接続後のスキーマビュー

SchemaAgent サーバーへの接続が確立された後は、スキーマビューの表示は以下のようになります:



メモ

- グローバルビューの上部には「SchemaAgent Server と接続」という文字列が表示され、接続を確立したサーバー名が表示されます。
- サーバー検索パス上にある全てのスキーマやスキーマコンストラクトへアクセスすることができるようになります。SchemaAgent スキーマはグローバル要素のような複合型を構成し、(**黒字で表示される**) アクティブなスキーマの下で単純型は **太字の青色のテキスト**で表示されます。.

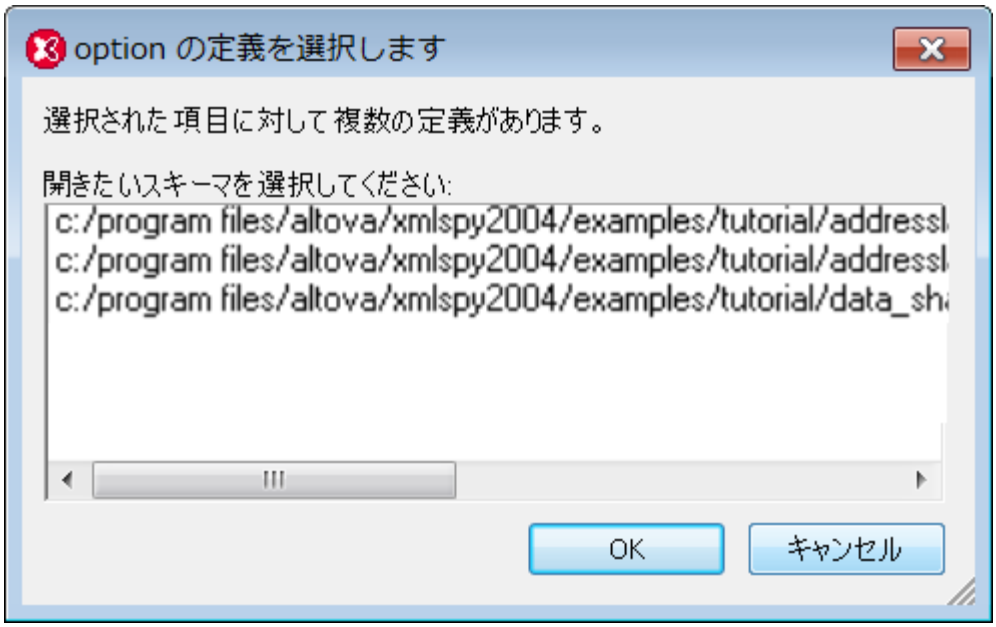


スキーマコンストラクトは、コンポーネント入力ヘルパーの対応するタブである型(グローバル)、名前空間、同一性制約から参照することができます。

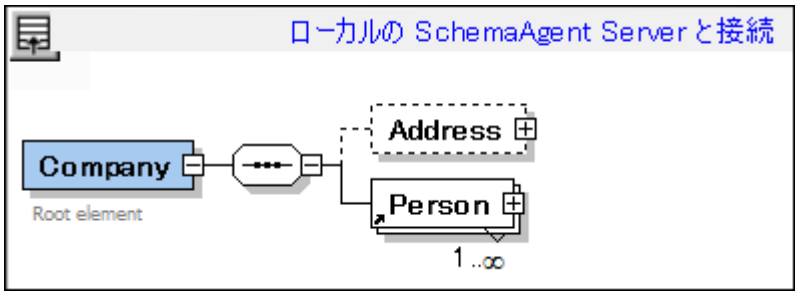
6.4.2 検索パス上のスキーマを開く

この例では、SchemaAgent Server で定義された検索パス上にあるスキーマを開く方法を説明します。ここでは `..\Tutorial` フォルダー以下にある `Db2schema.xsd` ファイルを、アクティブなスキーマとして使用します。コンポーネント入力ヘルパーのグローバルタブをアクティブにします。

1. コンポーネント入力ヘルパーにある青色で記された Company アイテムまで移動し、アイテムをダブルクリックします。定義へ移動ダイアログが開かれます。



2. Click the `Addresslast.xsd` エントリを選択して「OK」を押して確認します。これにより `MFCompany.xsd` スキーマが開かれ、`Company` 要素のコンテンツモデルが表示されます。



メモ　要素や複合型、または単純型といったSchemaAgent スキーマコンストラクトをダブルクリックすることで、関連するスキーマ(そしてインクルードされた他のスキーマ)がXMLSpy で開かれます。

6.4.3　IIR の使用

XML スキーマでは、Import、Include、そして Redefine (IIR) ステートメントを使用することでスキーマをモジュール化することができます。各メソッドでは異なる名前空間が必要になります。IIR を行う際に必要となる項目はIIR を作成する際にSchemaAgent Client と XMLSpy により自動的にチェックされます。

インポート(Import)、インクルード(Include)、そして再定義(Redefine) (IIR)

スキーマコンストラクトを「挿入」するにはいくつか方法があります:

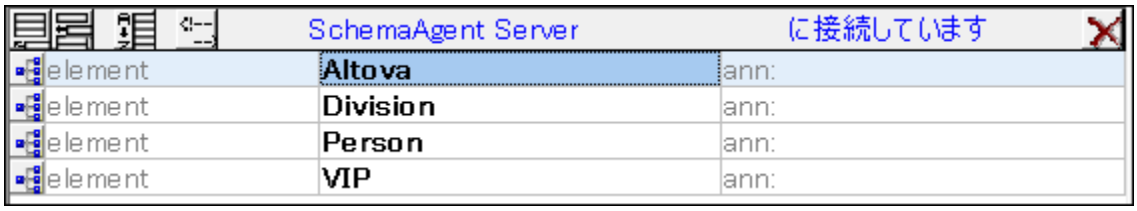
- グローバル要素はコンポーネント入力ヘルパーから、(スキーマビューの)スキーマコンポーネントのコンテンツモデルに直接ドラッグすることができます。
- 新規要素や属性などを定義する際に自動的に開かれるリストボックスから、複合型や単純型のようなコンポーネントを選択することができます。
- 複合型の様なコンポーネントを作成、更新する際に、詳細入力ヘルパーからコンストラクトを選択することができます。

スキーマコンポーネントの組み込み

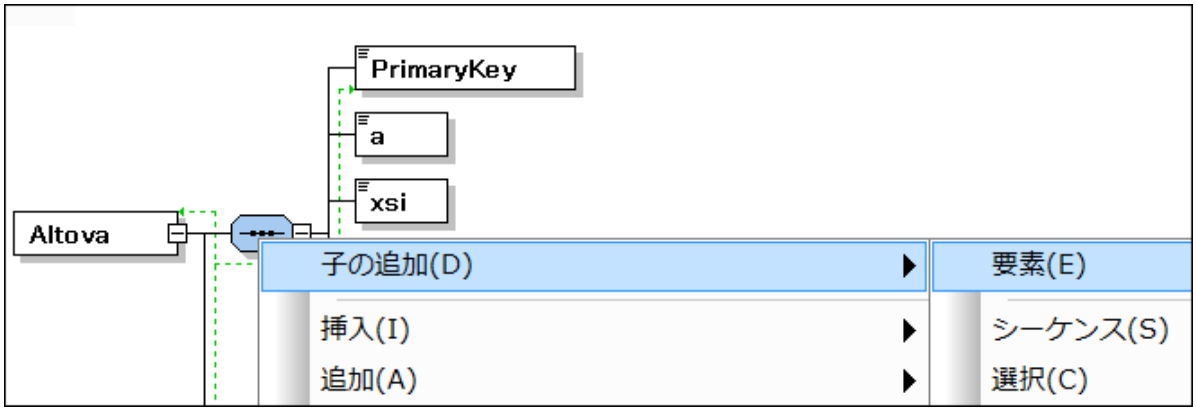
この例ではアクティブなスキーマとして..\Tutorial フォルダー内のDb2schema.xsd ファイルを使用します。コンポーネント入力ヘルパーグローバルが有効化されています。

SchemaAgent Server スキーマにあるスキーマコンストラクトを使用するには:

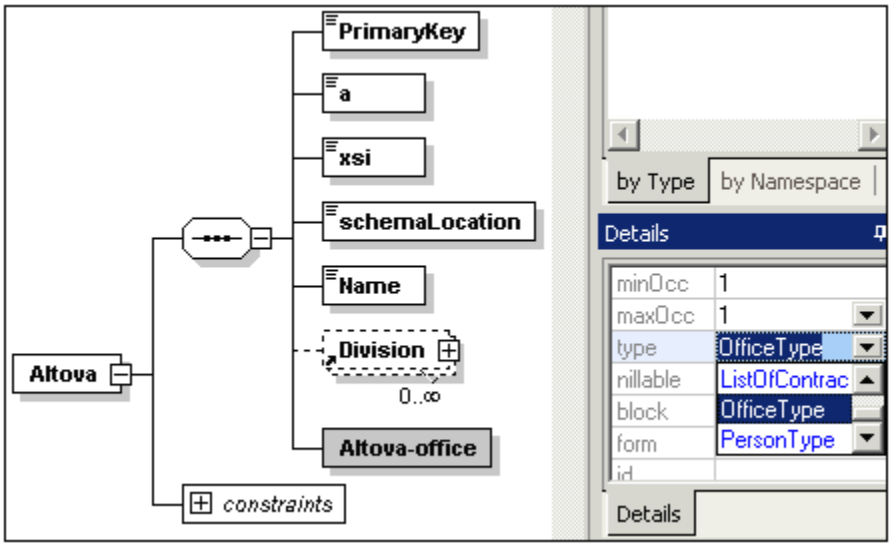
1. SchemaAgent サーバーに接続していることを確認します(SchemaAgent サーバーへ接続 290 を参4. Altova 以下にあるシーケンスコンポジッターを右クリックして、メニューオプションから「子を追加 | 要素」を選択します。照)。
2. DB2Schema.xsd ファイルを、この例のために(例えば) Altova-officeという名前に変更します。



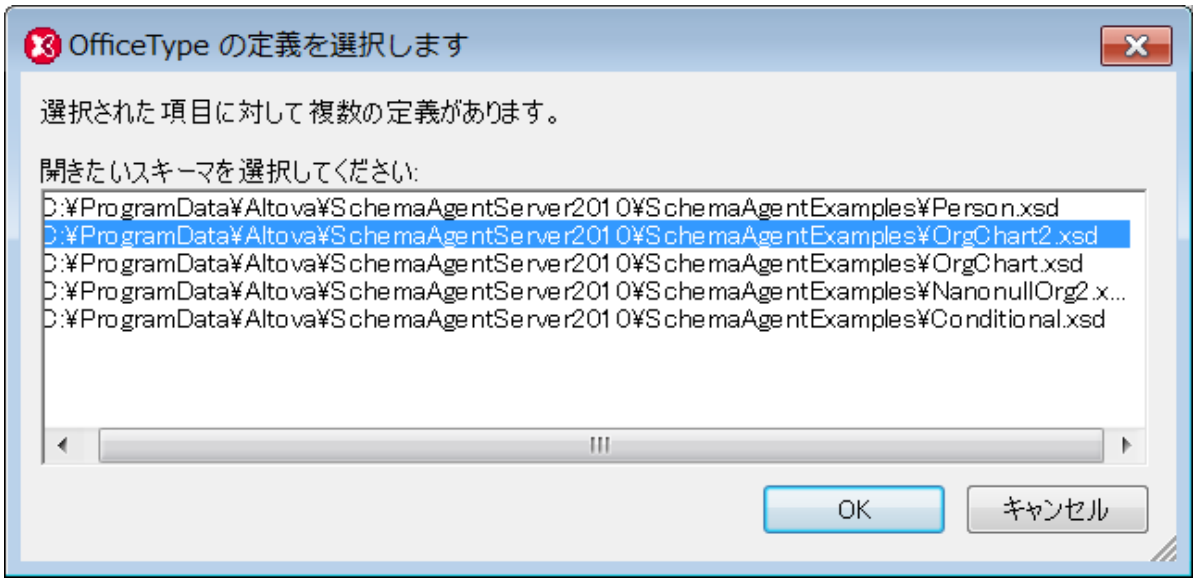
3. スキーマ概要に表示されている Altova 要素のアイコンをクリックして、コンテンツモデルを表示します。
4. Altova シーケンスコンポジッターを右クリックして、メニューオプション「子を追加 | 要素」を選択します。サーバーパス内のすべてのグローバル要素を含むリストボックスはこの時点で自動的に開かれます。1つ選択するとその要素は組み込まれます。



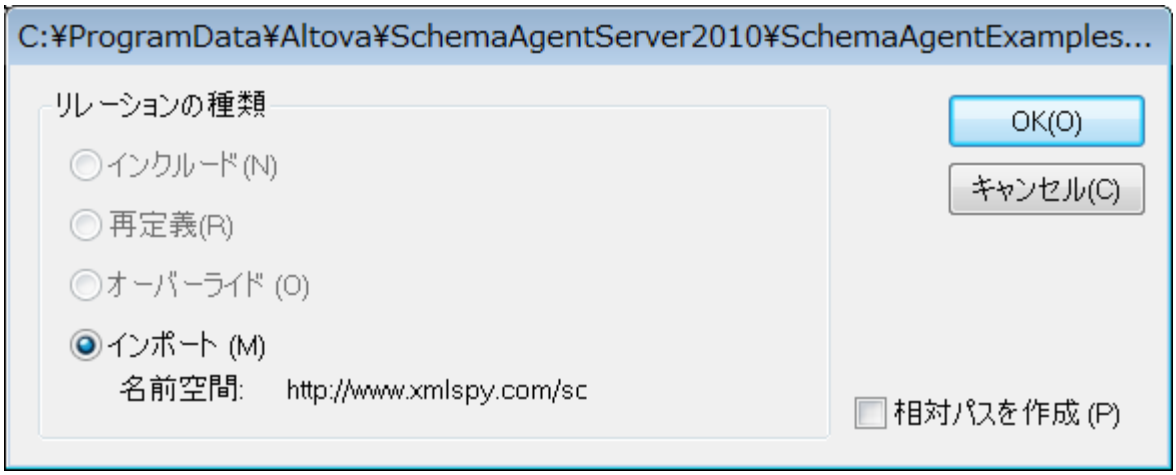
5. 新たに作成した要素の名前をAltova-office として、「Enter」で確定します。
6. 詳細入力ヘルパーにあるtype コンボボックスをクリックして、OfficeType を選択します。



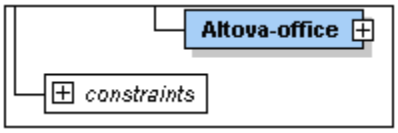
OfficeType の定義を選択するダイアログボックスが表れます。



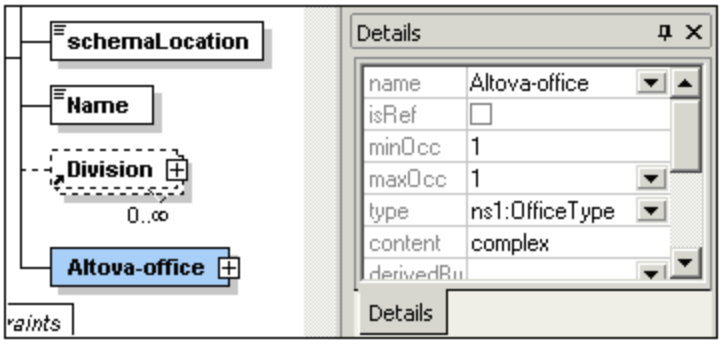
7. `Orgchart.xsd`を選択して「OK」をクリックして確認します。



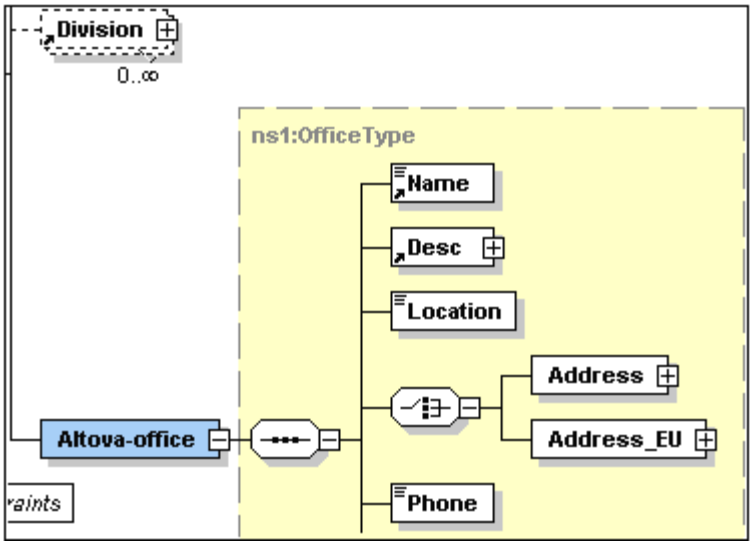
8. 「OK」をクリックします。インポートコマンドが自動的に選択されます。`Altova-office` 要素に展開アイコンが表示されます。



メモ `Orgchart.xsd` スキーマファイルがインポートされると、各スキーマのターゲット名前空間は異なるものと見なす必要があるため詳細入力ヘルパーの type エントリは `ns1:OfficeType` に変更されます。インポートコマンドもスキーマに追加されます。



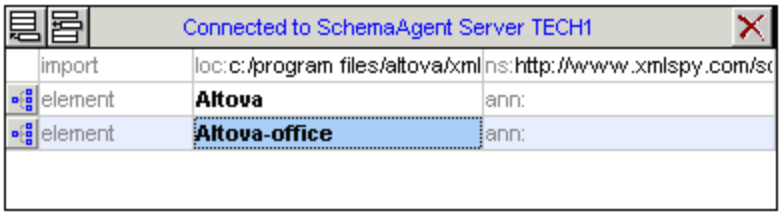
9. 展開ボタンをクリックして、`OfficeType` コンテンツモデルを確認します。



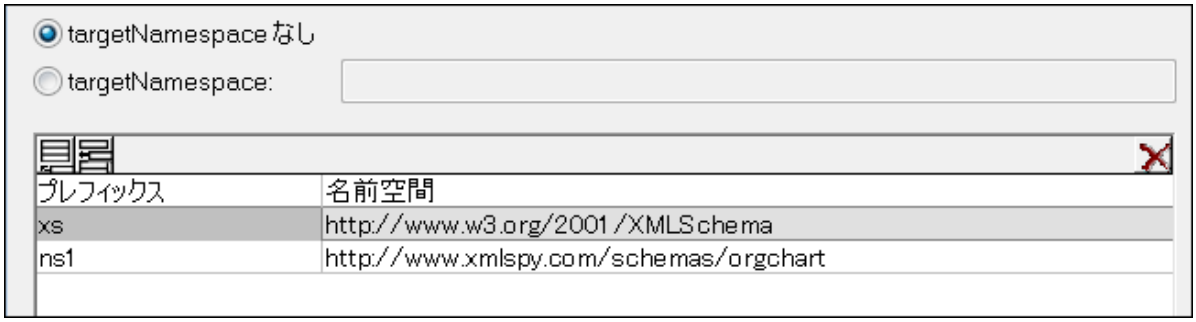
10. **F8** キーを押してスキーマを検証します。「スキーマは妥当です」というメッセージが表示されるはずです。

スキーマの整理:

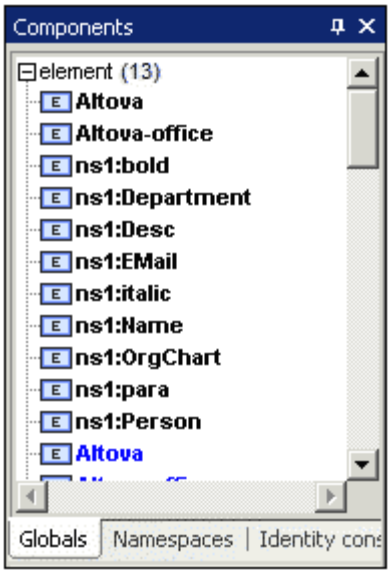
1. コンテンツモデルの `Division` 要素を削除します。
2. 概要へ戻るアイコン をクリックして、スキーマ概要へ画面を切り替えます。
3. 以下のグローバル要素を削除します: `Division`、`Person`、`VIP`。



4. メニューオプションから「**スキーマ設計 | スキーマ設定**」を選択し、名前空間の設定が変更されていることを確認します。



`ns1` プレフィックスが `www.xmlspy.com/schemas/orgchart` 名前空間に自動的に追加されます。コンポーネント（*スクリーンショット参照*）と詳細入力ヘルパーは `ns1:` namespace prefix.名前空間プレフィックスからインポートされた全てのコンストラクトを表示します。



メモ

- XMLSpyを使ってSchemaAgent Server コントロールを変更することで、SchemaAgent Server のパス上にある（変更されたスキーマを参照している）他のスキーマも自動的に更新されます。
- 作業中のスキーマがSchemaAgent Server パス上にある場合、入力ヘルパーで複製されたコンストラクト要素や単純型を（黒色と青色の違いにより）確認することも可能です。

6.4.4 SchemaAgent でスキーマを閲覧

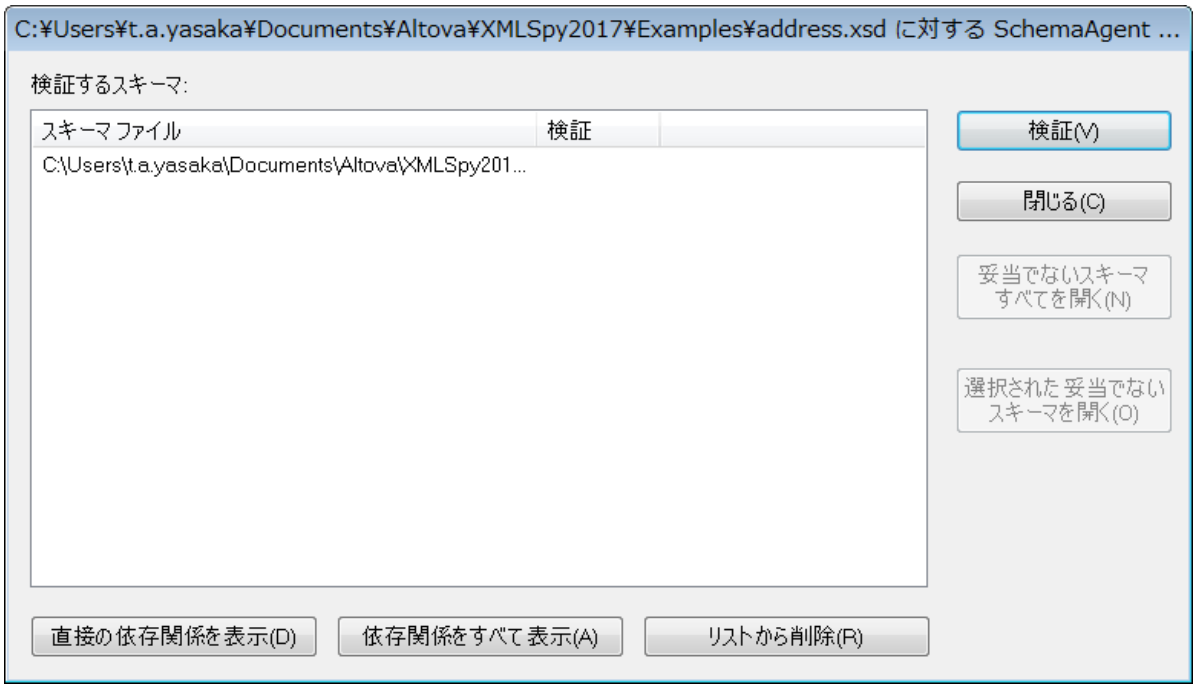
アクティブならびに関連したスキーマを SchemaAgent を使って確認、編集するには、メニューオプションから「**スキーマ設定 | SchemaAgent で表示 | *スキーマまたは関連するスキーマ***」を選択します（以下のスクリーンショットを参照）。
SchemaAgent を使って、アクティブなスキーマのみ（「**ファイルのみ**」コマンド）、またはアクティブなスキーマと、(i) 直接参照されている全てのスキーマか、(ii) 直接参照している全てのスキーマか、(iii) 直接関連する全てのスキーマを開くことができます。

6.4.5 SchemaAgent 検証

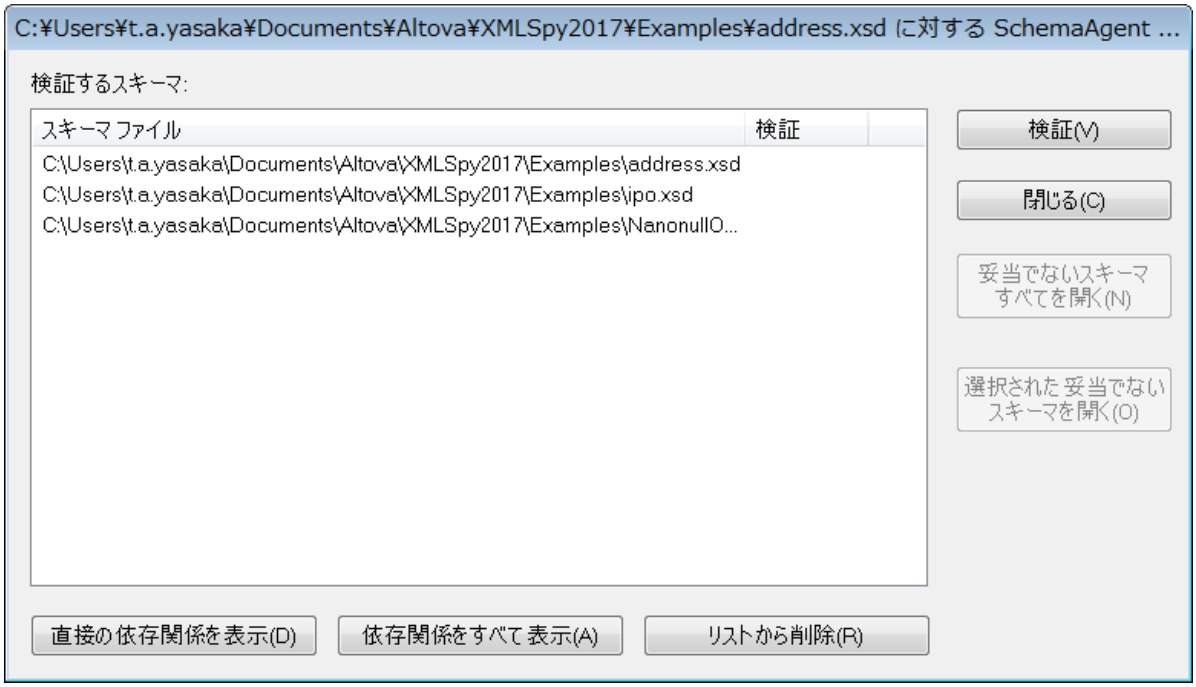
SchemaAgent と組み合わさった XMLSpy により、現在アクティブなスキーマのみならず、それに関連したスキーマの検証も行うことができます。このようなスキーマの検証は、SchemaAgent 検証と呼ばれます。関連するスキーマには (i) 直接の依存関係を持つスキーマ（直接参照している、または直接参照されているスキーマ）と、(ii) 全ての依存関係を持つスキーマ（直接の依存関係に加え、中間的なスキーマを介することで、他のスキーマに関連したスキーマセットの間接的な依存関係も含まれます）の 2 種類があります。

SchemaAgent 検証がどのように行われるかは、以下の例により示されます。この例では `address.xsd` というファイルが XMLSpy のスキーマビューのアクティブなスキーマであることを前提にしています。SchemaAgent 検証のコマンドを使用にするには、SchemaAgent Server の検索パスにアクティブなファイルと依存関係を持つファイルが含まれることを確認してください。以下の操作を行ってください：

1. ツールバーにある「**SchemaAgent 検証**」アイコン をクリックするか、メニューアイテムの「**スキーマ設計 | SchemaAgent 検証**」を選択します。この操作により SchemaAgent 検証ダイアログが開かれ、アクティブなスキーマのみ、または関連するスキーマまでも検証を行うか選択することができます（以下のスクリーンショットを参照）。

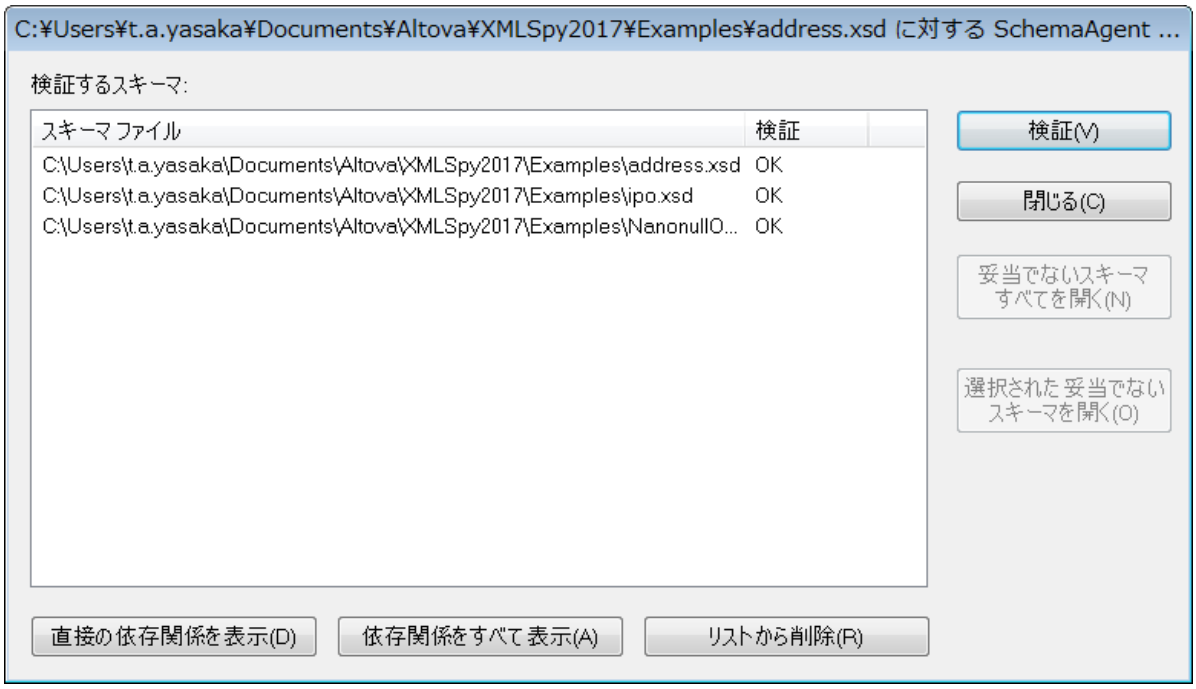


2. スキーマをリストに追加するには、「**直接の依存関係を表示**」または「**依存関係を全て表示**」ボタンをクリックします。この例では、「**依存関係を全て表示**」ボタンをクリックすることで、直接参照されているファイル、または間接的に参照されているファイルがリストに表示されます。



この時点で、必要に応じて(「**リストから削除**」により)スキーマをリストから取り除くことができます。

3. 「**検証**」ボタンをクリックして、リストボックスに表示されているスキーマ全ての検証を行ないます。



検証列に検証結果が表示されます。

妥当でないスキーマ全て、または選択されたうち妥当でないスキーマセットを XMLSpy で開くことができます。

6.5 XMLSpy 内のカタログ

XMLSpy はOASIS XML カタログメカニズムのサブセットをサポートします。XMLSpy のカタログ機能により、広く使われているスキーマ（またはスタイルシートなどその他のファイル）をローカルユーザーフォルダーから取得することができます。これにより全体的な処理スピードが向上し、（ネットワークに接続していない状態でも）ユーザーはオフラインで作業することができるようになり、（URI の変更はカタログファイル内部に限定されるので）ドキュメントのポータビリティーが向上します。

XMLSpy のカタログ機能は以下のように動作します。

- カタログの仕組み 301
- XMLSpy 内のカタログ構造 302
- カタログのカスタム化 303
- 環境変数 305

カタログの詳細に関しては XML カタログ仕様 を参照してください。

6.5.1 カタログのしくみ

カタログをDTD とXML スキーマをリダイレクトするために使用することができます。両方のケースのメカニズムの後ろにあるコンセプトは同じですが、違いの詳細は以下に説明されている通りです。

DTD

カタログはDTD への呼び出しをローカルのURI にリダイレクトするために使用されます。これはマッピング、カタログファイル、ローカルURI により必要とされるパブリックまたはシステム識別子により達成されます。XML ファイル内のDOCTYPE 宣言が読み取れれると、パブリックまたはシステム識別子が必要とされるローカルリソースをカタログファイルマッピングによりロケートします。

人気のあるスキーマに関してはPUBLIC 識別子は通常定義済みです。このためカタログファイル内のURI のみが正しいローカルコピーに対してPUBLIC 識別子 をマップします。XML ドキュメントが解析されると内部のPUBLIC 識別子が読み取られます。この識別子がカタログファイル内で見つかると、カタログファイル内の対応するURL がルックアップされ、スキーマがこのロケーションから読み取られます。例えば、次のSVG ファイルがXMLSpy 内で開かれるとします:

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
"http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">

<svg width="20" height="20" xml:space="preserve">
  <g style="fill:red; stroke:#000000">
     <rect x="0" y="0" width="15" height="15"/>
     <rect x="5" y="5" width="15" height="15"/>
  </g>
</svg>
```

カタログはこのSVG ファイルのPUBLIC 識別子を検索します。カタログファイルは以下のエントリを含んでいます:

```
<catalog>
  ...
    <public publicId="-//W3C//DTD SVG 1.1//EN" uri="schemas/svg/svg11.dtd"/>
  ...
</catalog>
```

この場合 **PUBLIC** 識別子のために一致するものがあります。この結果 SVG DTD のためのルックアップが(カタログファイルに相対的な) URL **schemas/svg/svg11.dtd** にリダイレクトされます。これはSVG ファイルのためにDTD として使用されるローカルファイルです。カタログ内で **Public** ID のためのマッピングが存在しない場合、XML ドキュメント内のURL が使用されます(上のSVG ファイルのサンプルの場合、これはインターネット URL です: **http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd**)。

XML スキーマ

XMLSpy 内では XML スキーマをリダイレクトするためにカタログを使用することができます。XML インスタンスファイル内で、スキーマへのレファレンスがXML ドキュメントのトップレベルの要素の **xsi:schemaLocation** 属性内で発生します。例えば、

```
xsi:schemaLocation="http://www.xmlspy.com/schemas/orgchart OrgChart.xsd"
```

xsi:schemaLocation 属性の値には2 つの部分があります: (上で緑色で表示されている)名前空間のパートと(ハイライトされている) URI パート。名前空間の部分は代替リソースをマップするためにカタログ内で使用されています。例えば、以下のカタログエントリは上のスキーマレファレンスを代替のロケーションのスキーマにリダイレクトします。

```
<uri name="http://www.xmlspy.com/schemas/orgchart" uri="C:\MySchemas\OrgChart.xsd"/>
```

通常 **xsi:schemaLocation** 属性の値のURI 部分は実際のスキーマのロケーションのパスです。しかしながら、スキーマがカタログを介して参照される場合、URI の部分は実際のXML スキーマをポイントする必要はなく、**xsi:schemaLocation** 属性の文法の妥当性が保持されるために存在する必要があります。`Foo` の値は、例えば、属性の値の部分のが妥当であるために十分です。

6.5.2 XMLSpy 内のカタログ構造

XMLSpy が開始されると、(*以下のリスティングで示されるような構造を持った*) **RootCatalog.xml** というファイルがロードされ、ファイル内部に記述されているカタログファイルが検索されます。このファイルを修正して、検索するカタログファイルを増やすこともできます(各カタログファイルは **nextCatalog** 要素にて記述されます)。各カタログファイル内に記述されているURI が、ファイル内で指定されたマッピングに従って解決されます。

Listing of RootCatalog.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<catalog xmlns="urn:oasis:names:tc:entity:xmlns:xml:catalog"
   xmlns:spy="http://www.altova.com/catalog_ext"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="urn:oasis:names:tc:entity:xmlns:xml:catalog Catalog.xsd">
  <nextCatalog catalog="%PersonalFolder%/Altova/%AppAndVersionName%/CustomCatalog.xml"/>
  <!-- Include all catalogs under common schemas folder on the first directory level -->
  <nextCatalog spy:recurseFrom="%CommonSchemasFolder%" catalog="catalog.xml"
spy:depth="1"/>
  <nextCatalog spy:recurseFrom="%ApplicationWritableDataFolder%/pkgs/.cache"
catalog="remapping.xml" spy:depth="0"/>
  <nextCatalog catalog="CoreCatalog.xml"/>
</catalog>
```

上のリスティングはカスタムカタログ(named **CustomCatalog.xml**) と(W 3C XML スキーマとSVG スキーマ)などの一般的に使用されるスキーマをロケートするカタログのセットを参照します

- (**%PersonalFolder%** を介してロケートされる) **CustomCatalog.xml** は個人用フォルダーにあります自身のマッピングを作成するスケルトンファイルです。共通のスキーマフォルダー内のカタログファイルにより指されていない必要とするスキーマのためにマッピングを **CustomCatalog.xml** に追加することができます。OASIS カタログメカニズムのサポートされる要素を使用してこれを行います(*下を参照*)。

(C) 2015-2021 Altova GmbH

- （共通に使用されるスキーマのセットを含む変数 **%CommonSchemasFolder%** を使用してロケートされる）共通のスキーマフォルダーこれらのスキーマフォルダの内部は保存された対応するスキーマのコピーをポイントするURI に対するパブリックおよび/またはシステム識別子をマップする **catalog.xml** ファイルです。
- **CoreCatalog.xml** はXMLSpy アプリケーションフォルダー内にあり、XML ドキュメントのAltova のAuthentic ビューを生成するために使用されるStyleVision パワースタイルシートなどXMLSpy-固有のプロセスにより使用されるスキーマとスタイルシートをロケートするために使用されます。

ロケーション変数
RootCatalog.xml 内で使用されている変数は以下の値を持っています（*上のリスティング*）:

%PersonalFolder%	現在のユーザーの個人的なフォルダー、例えば C:\Users\<name>\Documents
%CommonSchemasFolder%	C:\ProgramData\Altova\Common2021\Schemas
%ApplicationWritableDataFolder%	C:\ProgramData\Altova

カタログファイルとスキーマのロケーション
多種のカタログファイルの場所に注意してください。

- **RootCatalog.xml** と**CoreCatalog.xml** はXMLSpy アプリケーションフォルダー内にあります。
- **CustomCatalog.xml** は**MyDocuments\Altova\XMLSpy** フォルダー内にあります。
- **Catalog.xml** 入るはそれぞれ特定のスキーマフォルダー内にあり、これらのスキーマフォルダーは共通のスキーマフォルダー内にあります。

6.5.3 カタログのカスタム化

CustomCatalog.xml（またはXMLSpy により読み取られる他のカタログファイル）内でエントリを作成する場合OASIS カタログ仕様の以下の要素のみを使用してください。下の各要素は属性の値の説明と共にリストされています。詳細に関しては（XML カタログ仕様）を参照してください。各要素はその要素のベースURI を指定するために使用される **xml:base** 属性を取ることができます。

- `<public publicId="PublicID of Resource" uri="URL of local file"/>`
- `<system systemId="SystemID of Resource" uri="URL of local file"/>`
- `<uri name="filename" uri="URL of file identified by filename"/>`
- `<rewriteURI uriStartString="StartString of URI to rewrite" rewritePrefix="String to replace StartString"/>`
- `<rewriteSystem systemIdStartString="StartString of SystemID" rewritePrefix="Replacement string to locate resource locally"/>`

以下の点に注意してください：

- 多くのスタイルシート同様パブリック識別子が存在しない場合、システム識別子を **system** 要素を介して直接 URL にマップすることができます
- uri 要素を使用してURI を他のURI にマップすることができます。
- rewriteURI とrewriteSystem 要素によりURI またはシステム識別子の開始の部分をそれぞれ書き換えることが可能になります。これによりファイル パスの開始部分が置き換えられこの結果他のディレクトリをターゲットにすることができます。これらの要素の詳細に関してはXML カタログ仕様 を参照してください。

リリース2014 以降XMLSpy はXML Catalogs specification (OASIS Standard V1.1, 7 October 2005) 仕様を厳密に順守しています。この仕様は厳密にパブリックID またはシステムID）外部識別子ルックアップを（パブリックID またはシステムID を持たな

いURI) URI ルックアップと区別しています。名前空間 URI は単純にパブリックID またはシステムID ではなく URI と考えられ外部識別子ルックアップではなく URI ルックアップとして使用されるべきです。バージョン 2014 以前のXMLSpy バージョンではスキーマ名前空間 URI は**<public>** マッピングを使用して訳されてきました。2014 以降から**<uri>** マッピングが使用されています。

```
v2014 以前： <public publicID="http://www.MyMapping.com/ref"
   uri="file:///C:/MyDocs/Catalog/test.xsd"/>
V-2014 以降: <uri name="http://www.MyMapping.com/ref"
   uri="file:///C:/MyDocs/Catalog/test.xsd"/>
```

XMLSpy が参照されるスキーマを検索する方法

xsi:scemaLocation 属性を介して XML ドキュメント内で参照されるスキーマ（*以下に表示されています*）。**xsi:schemaLocation** 属性の値には2 つの部分があります: （緑色で表示されている）名前空間のパートと（ハイライトされている）URI パート。

```
xsi:schemaLocation="http://www.xmlspy.com/schemas/orgchart OrgChart.xsd"
```

以下のステップはXMLSpy がシーケンスにより参照されたスキーマを検索するステップを示しています。スキーマは最初のステップでロードされます。

1. **xsi:schemaLocation** 値のURI 部分のためのカタログをルックアップします。**rewriteURI** マッピング内を含みマッピングが検索されるとスキーマのロードのために結果のURI を使用します。
2. **xsi:schemaLocation** 値の名前空間部分のためのカタログをルックアップします。**rewriteURI** マッピング内を含みマッピングが検索されるとスキーマのロードのために結果のURI を使用します。
3. スキーマのロードのために**xsi:schemaLocation** 値のURI 部分を使用します。

スキーマ別のファイル拡張子とインテリジェントな編集

カタログファイルを介して特定のファイル拡張子を持つドキュメントがXMLSpy のインテリジェントな編集機能を指定するスキーマ内のルールに順守するように適用することができます。例えば、HTML DTD に従い妥当な（HTML）ファイルのためにカスタムファイル拡張子 `.myhtml` を作成する場合、**CustomCatalog.xml** を**<catalog>** 要素の子と追加してこれらの拡張子を持つファイルのためにインテリジェントな編集を有効化することができます。

```
<catalog>
  ...
   <spy:fileExtHelper ext="myhtml" uri="schemas/xhtml/xhtml1-transitional.dtd"/>
  ...
</catalog>
```

これはXMLSpy 内の **.myhtml** ファイルのインテリジェントな編集（自動補完、エントリヘルパーなど）をXHTML 1.0 Transitional DTD に従い有効化します。類似したエントリを含む**%AltovaCommonSchemasFolder%\Schemas\xhtml** フォルダー内の **catalog.xml** ファイルを参照してください。

XML スキーマ仕様

XML スキーマ仕様情報はXMLSpy にビルトされておりXML スキーマ（`.xsd`）ドキュメントの妥当性はこの内部情報に対してチェックされます。XML スキーマドキュメント内では、このためレファレンスはXML スキーマ仕様を定義するスキーマになってはなりません。

`%AltovaCommonSchemasFolder%\Schemas\schema` フォルダー内の**catalog.xml** ファイルは古いXML スキーマ仕様を実装するDTD へのレファレンスを含んできます。XML スキーマドキュメントをこれらのスキーマに対して検証してはなりません。参照されるファイルはドキュメントをこれらの古い（奨励に従い）作成する場合に編集の目的でXMLSpy に入力ヘルパー情報を与えるために含まれています。

6.5.4 環境変数

nextCatalog 要素ではシェル環境変数が使用され、システムに関係する場所へのパスが参照されます（*上の RootCatalog.xml の内容を参照*）。以下のシェル環境変数がサポートされます:

%PersonalFolder%	現在のユーザーの個人的なフォルダーへのフルパス、例えば C:\Users\<name>\Documents
%CommonSchemasFolder%	C:\ProgramData\Altova\Common2021\Schemas
%ApplicationWritableDataFolder%	C:\ProgramData\Altova
%AltovaCommonFolder%	C:\Program Files\Altova\Common2021
%DesktopFolder%	現在のユーザーのデスクトップフォルダーへのフルパス。
%ProgramMenuFolder%	現在のユーザーのプログラムメニューフォルダーへのフルパス。
%StartMenuFolder%	現在のユーザーの開始メニューフォルダーへのフルパス。
%StartUpFolder%	現在のユーザーのスタートフォルダーへのフルパス。
%TemplateFolder%	現在のユーザーのテンプレートフォルダーへのフルパス。
%AdminToolsFolder%	現在のユーザーの管理ツールを保管するファイルシステムディレクトリへのフルパス。
%AppDataFolder%	現在のユーザーのアプリケーションデータフォルダーへのフルパス。
%CommonAppDataFolder%	全てのユーザーのアプリケーションデータを含むファイルディレクトリへのフルパス。
%FavoritesFolder%	現在のユーザーの[お気に入り]フォルダへフルパス。
%PersonalFolder%	現在のユーザーの個人フォルダーへのフルパス。
%SendToFolder%	現在のユーザーの送信フォルダーへのフルパス。
%FontsFolder%	システムフォントフォルダーへのフルパス。
%ProgramFilesFolder%	現在のユーザーのプログラムファイルフォルダーへのフルパス。
%CommonFilesFolder%	現在のユーザーの共通ファイルフォルダーへのフルパス。
%WindowsFolder%	現在のユーザーの Windows フォルダーへのフルパス。
%SystemFolder%	現在のユーザーのシステムフォルダーへのフルパス。

%LocalAppDataFolder%	ローカル（非ローミング）アプリケーションのためのデータレポジトリとしての役割を果たすファイルシステムディレクトリへのフルパス。
%MyPicturesFolder%	ピクチャフォルダーへのフルパス。

6.6 スキーマの中を検索

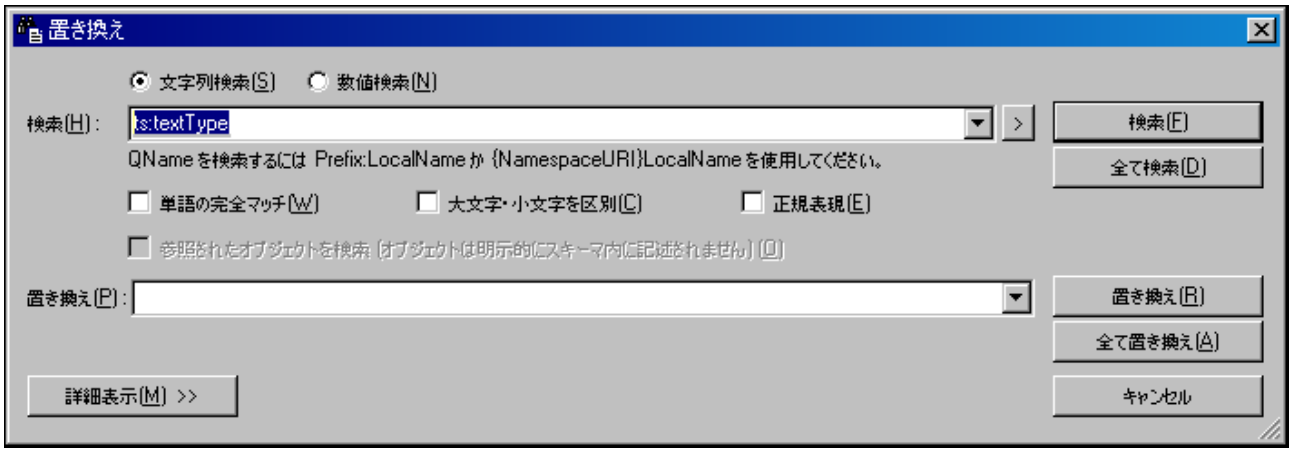
XMLSpy の検索と置き換え機能をスキーマビューで使うことで、インテリジェントな検索を行うことができます。

スキーマビューの検索と置き換え機能は、スキーマファイルがスキーマビューにてアクティブになっている時に使用することができます。以下の2つの方法でこの機能にアクセスすることができます:

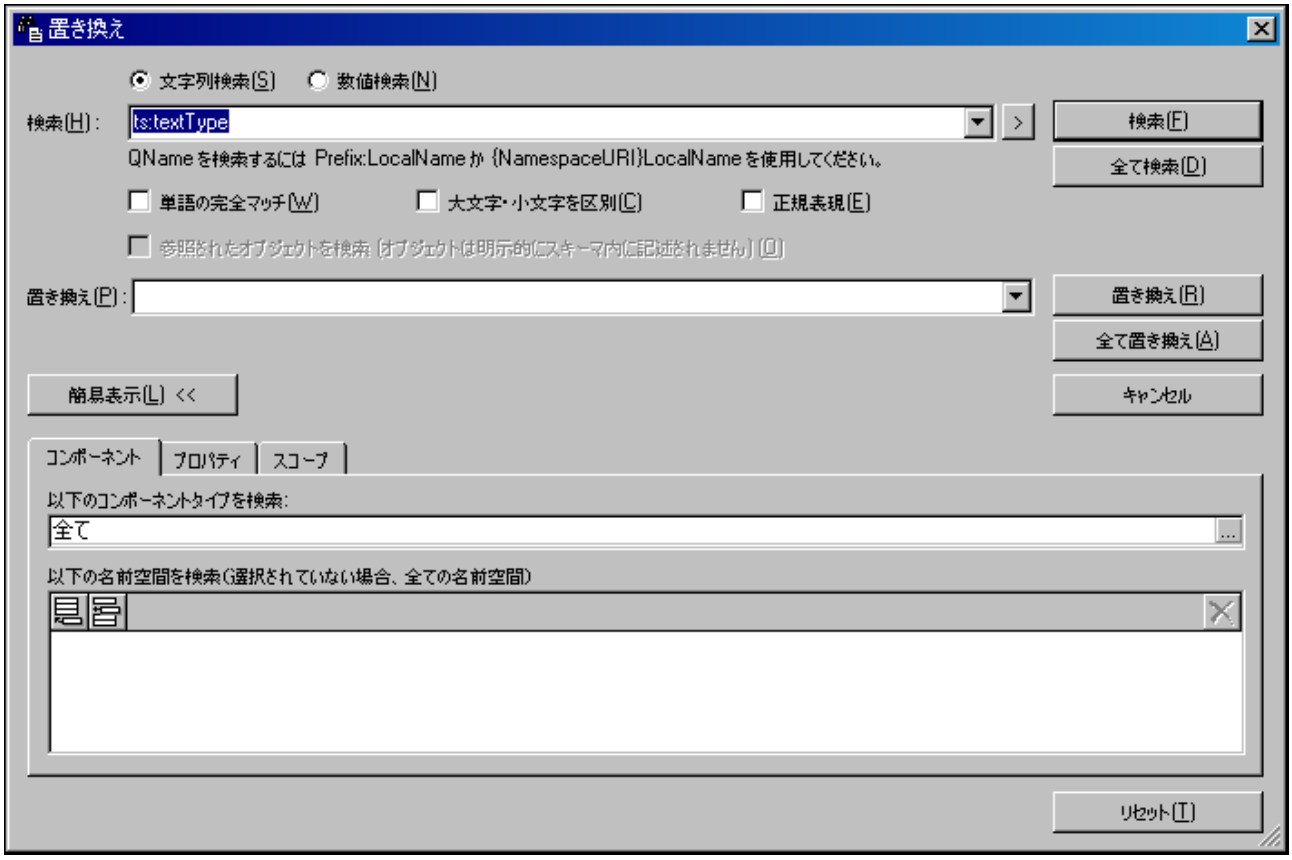
- メニューコマンドから「**編集 | 検索**」や「**編集 | 置換**」を選択する
- スキーマの中を検索ウィンドウにある「**検索**」または「**置き換え**」ボタンを押下する

コマンドを選択またはボタンをクリックすることで、検索または置き換えダイアログが表示されます。検索ダイアログと比べると、(*以下のスクリーンショットにある*) 置き換えダイアログには置き換えを行うテキストを表示するテキストフィールドが追加されています。

デフォルトの置き換えダイアログは以下のように表示されます:



「**詳細表示**」ボタンをクリックすることで、(*以下のスクリーンショットにあるように*) より詳細な検索を行うための条件が表示されます。



使用方法は以下のとおりです:

- 検索ならびに置き換えテキストフィールドにて検索、置き換えを行う単語を入力 309
- コンポーネントタブにて検索するスキーマコンポーネントを指定 310
- 検索するコンポーネントのプロパティを指定 311 して検索の対象を狭める
- 現在のドキュメント、プロジェクト、またはフォルダーを指定して検索の範囲をセット 314
- コマンドの実行 315
- スキーマの内を検索ウィンドウを使用 317 してコンポーネントの素早いナビゲーションを行う

ダイアログ下部にある「リセット」ボタンにより、以下にあるようなデフォルト状態に設定を戻します:

- 検索語無し、置き換え語無し
- コンポーネント: 全て
- 名前空間: 選択無し
- プロパティの制約: どこでも
- プロパティの追加制約: 無し
- スコープ: 現在のファイル

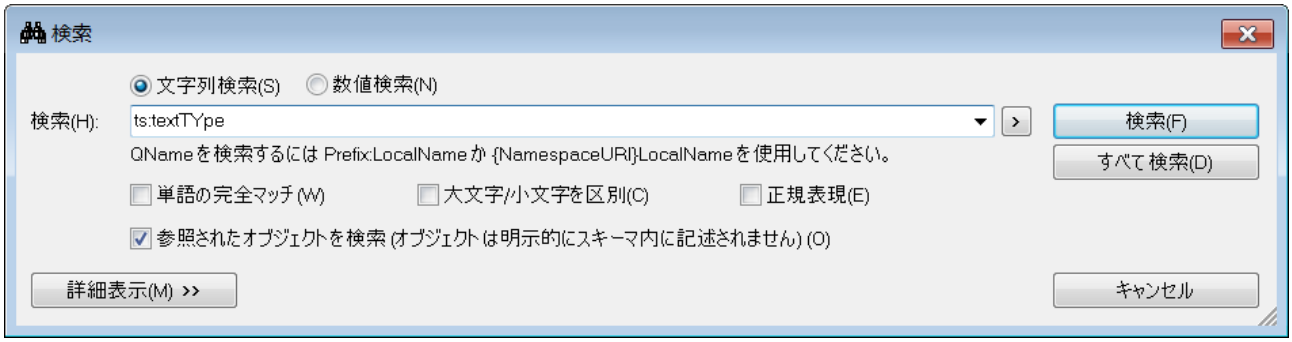
メモ 置換フィールド内で正規表現はサポートされていません。

6.6.1 検索語

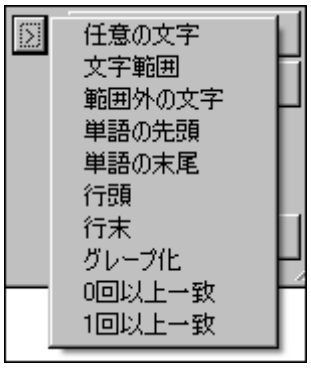
検索語は文字列（**文字列ラジオボタン**）または数値（**数値ラジオボタン**）として入力することができます。

文字列検索

文字列検索内で（*下のスクリーンショット*）エントリは以下であることができます：（I）テキスト（ii）QName または（iii）正規表現QName を使った検索では、テキストフィールドに入力されたドキュメントで使用されているプレフィックス、もしくは名前空間 URI をベースに名前空間が決定されます。以下のスクリーンショットでは、`ts:` プレフィックスが名前空間を決定するためにドキュメント内で使用されているプレフィックスになります。



正規表現を使った検索を行うには、正規表現チェックボックスにチェックを入れて、テキストフィールドに正規表現を入力してください。（*以下のスクリーンショットに示される*）検索語入力フィールドの右にある矢印ボタンをクリックすることで表示されるメニューから正規表現の入力を行うことができます。

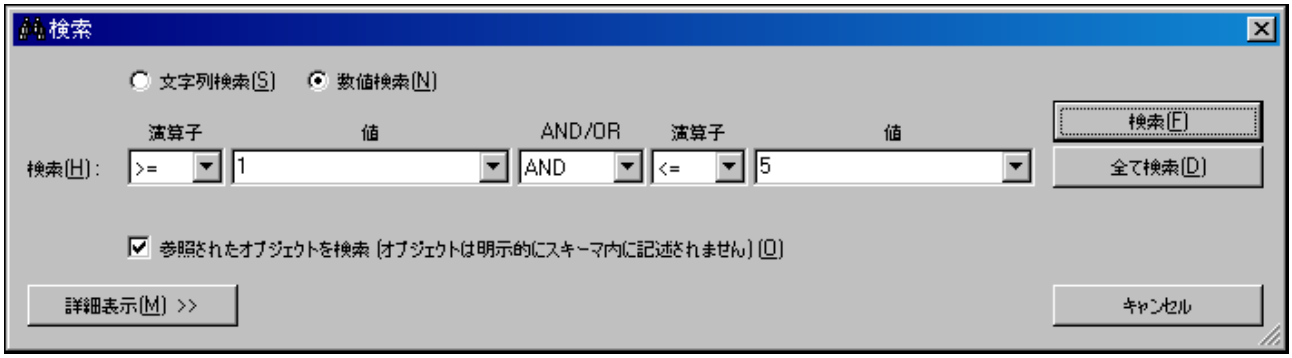


検索語がドキュメント内の単語に完全一致するか、または大文字/小文字を区別した検索を行うか設定することができます。テキストフィールドの下にあるチェックボックスを使って、これらのオプションを使用してください。

（複合型の定義やグローバル要素といった）参照されているオブジェクト内で検索を行ないたい場合は、参照されたオブジェクトを検索チェックボックスのチェックを入れてください。このオプションは検索ダイアログだけで使用できるもので、置き換えダイアログでは無効化されます。

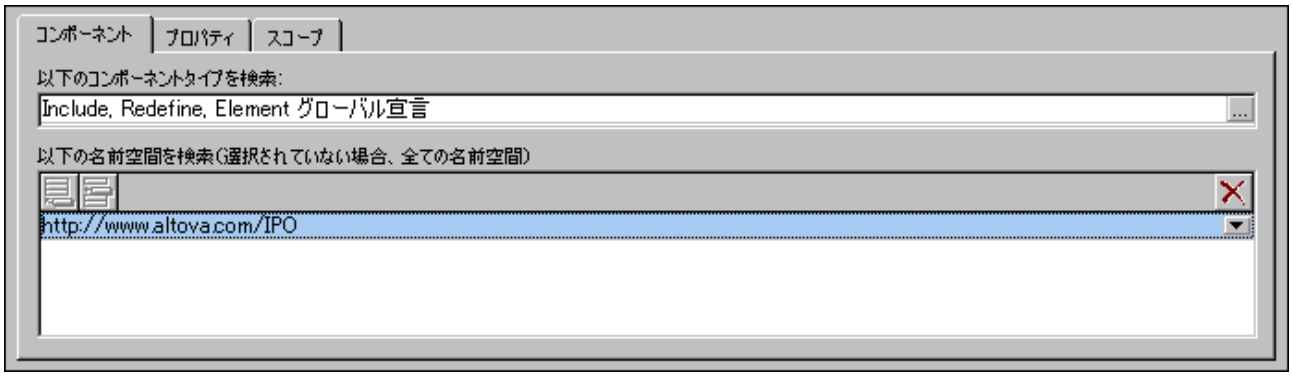
数値検索

数値検索ラジオボタンが選択されていれば、1 つの演算子と数値からなる検索条件か、2 つの検索条件を `AND` または `OR` の論理式により組み合わせることで検索を行うことができます。以下のスクリーンショットでは、2 つの検索パラメーターにより、「1 以上かつ 5 以下」という検索条件が指定されています。



6.6.2 コンポーネント

コンポーネントの種類やターゲット名前空間により検索を行う対象を制限することができます。これらのオプションはコンポーネントタブから利用することができます。検索または置き換えダイアログにて「**詳細表示**」ボタンをクリックしてください。検索の対象を狭めるタブが開かれ、そのうちの1つが(*以下のスクリーンショットに示される*)コンポーネントタブになります。

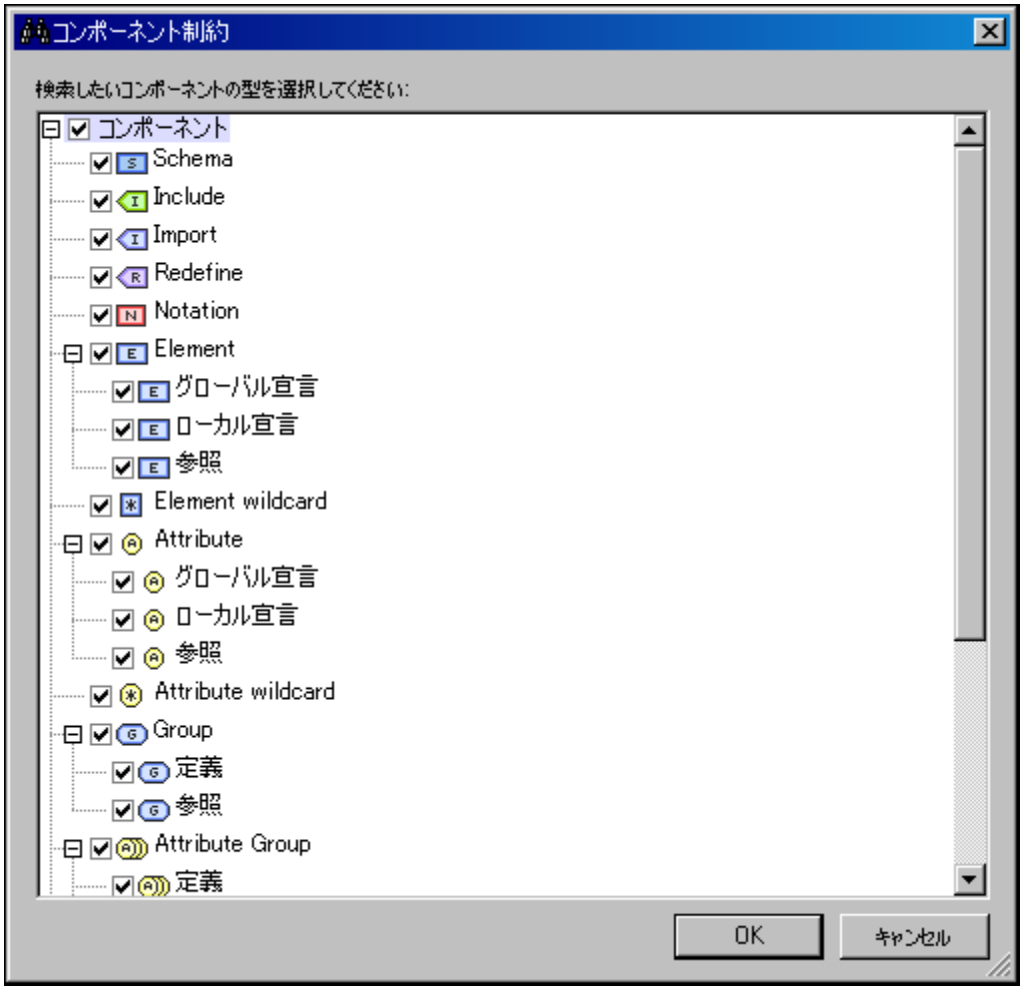


コンポーネントタブは2つの部分から構成されています: コンポーネントタブは、(i) 検索を行うコンポーネントの種類と、(ii) 検索を行うターゲット名前空間という2つの部分から構成されます。

コンポーネントの選択

テキストフィールドの右側にある **追加** アイコンをクリックすることにより検索されるコンポーネント型を入力することができます。 ... (*上のスクリーンショットを参照*)。この操作により(*以下のスクリーンショットに示される*)コンポーネント制約ダイアログが表示され、クリック操作で検索を行う対象コンポーネントの種類を選択することができます。リストの上の `Components` アイテムをチェックすると全てのコンポーネントが選択されます(テキストエントリ `all`)。チェックを解除すると選択されたコンポーネントのそれぞれが含まれる全てのコンポーネント選択が解除されます(テキストエントリ `none`)—。従って、各コンポーネントを個々に選択するには、コンポーネント アイテムの選択が解除されていなければなりません。選択されたコンポーネントはカンマ区切りでテキストフィールドに表示されます(*上のスクリーンショットを参照*)。

(C) 2015-2021 Altova GmbH



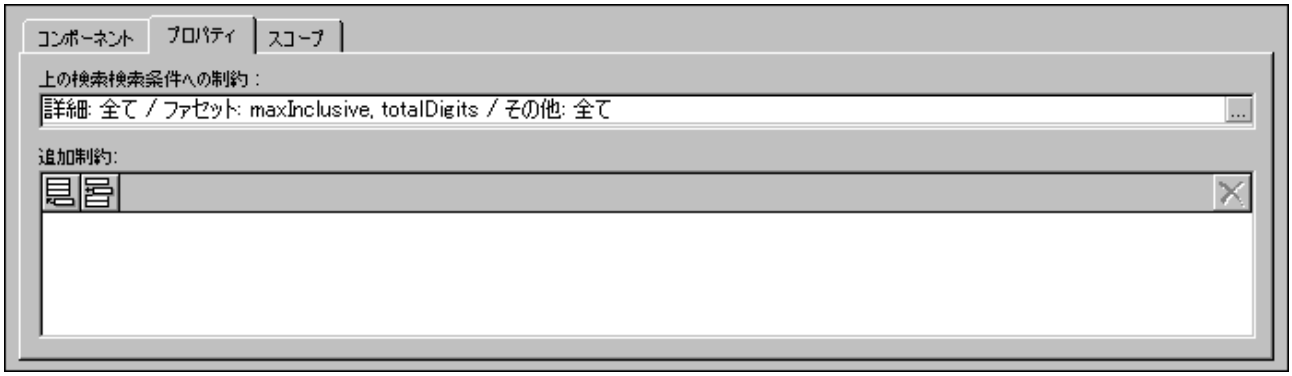
注意:　コンポーネントタブまたは検索、置き換えダイアログが再度開かれたとしても、前回のコンポーネント選択状態は保持されたままになります。

名前空間の選択

検索の対象とするターゲット名前空間を設定するには、「**追加**」または「**挿入**」アイコンをクリックして、目的の名前空間を入力してください。ターゲット名前空間が指定されていない場合、全てのターゲット名前空間が選択されます。ペインに入力されたターゲット名前空間を削除するには、そのターゲット名前空間を選択して、「**削除**」アイコンをクリックしてください。

6.6.3　プロパティ

プロパティタブのオプションを使用することで、コンポーネントプロパティ(詳細ならびにファセット)や、プロパティの内容で検索結果を絞り込むことができます。「**詳細表示**」ボタンをクリックすることで検索または置き換えダイアログが展開表示されるので、プロパティタブを選択します(*以下のスクリーンショットを参照*)。



プロパティタブは2つの部分から構成されています: (i) (メインの検索テキストフィールドに表示される) 検索対象の制約と、(ii) コンテンツ制約 (*追加の制約* 313 *を参照してください。*)

プロパティの選択

テキストフィールドの右側にある **追加** アイコンをクリックすることにより検索されるプロパティ型を入力することができます。 ... (*上のスクリーンショットを参照*)。この操作により(*以下のスクリーンショットに示される*) プロパティ制約ダイアログが表示され、クリック操作で検索を行う対象コンポーネントの種類を選択することができます。プロパティ制約ダイアログが表示され、アイテムをクリックすることで検索を行うプロパティを選択することができます。プロパティは(i) 詳細、(ii) ファセット、(iii) その他の3 つのグループに分類されています。詳細、ファセット、またはその他をクリックすることで、グループにある全てのプロパティが選択されます。グループを再度クリックすると、そのグループのプロパティ全てが(個別に選択されたプロパティも含めて) 選択解除された状態になります。個々のプロパティはグループアイテムが選択されていない状態でないと選択することができません。選択されたコンポーネントはカンマ区切りでテキストフィールドに表示されます(*上のスクリーンショットを参照*)。

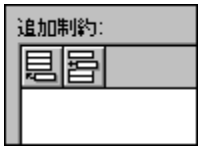
注意: プロパティタブまたは検索、置き換えダイアログが再度開かれたとしても、前回のプロパティ選択状態は保持されたままになります。

(C) 2015-2021 Altova GmbH

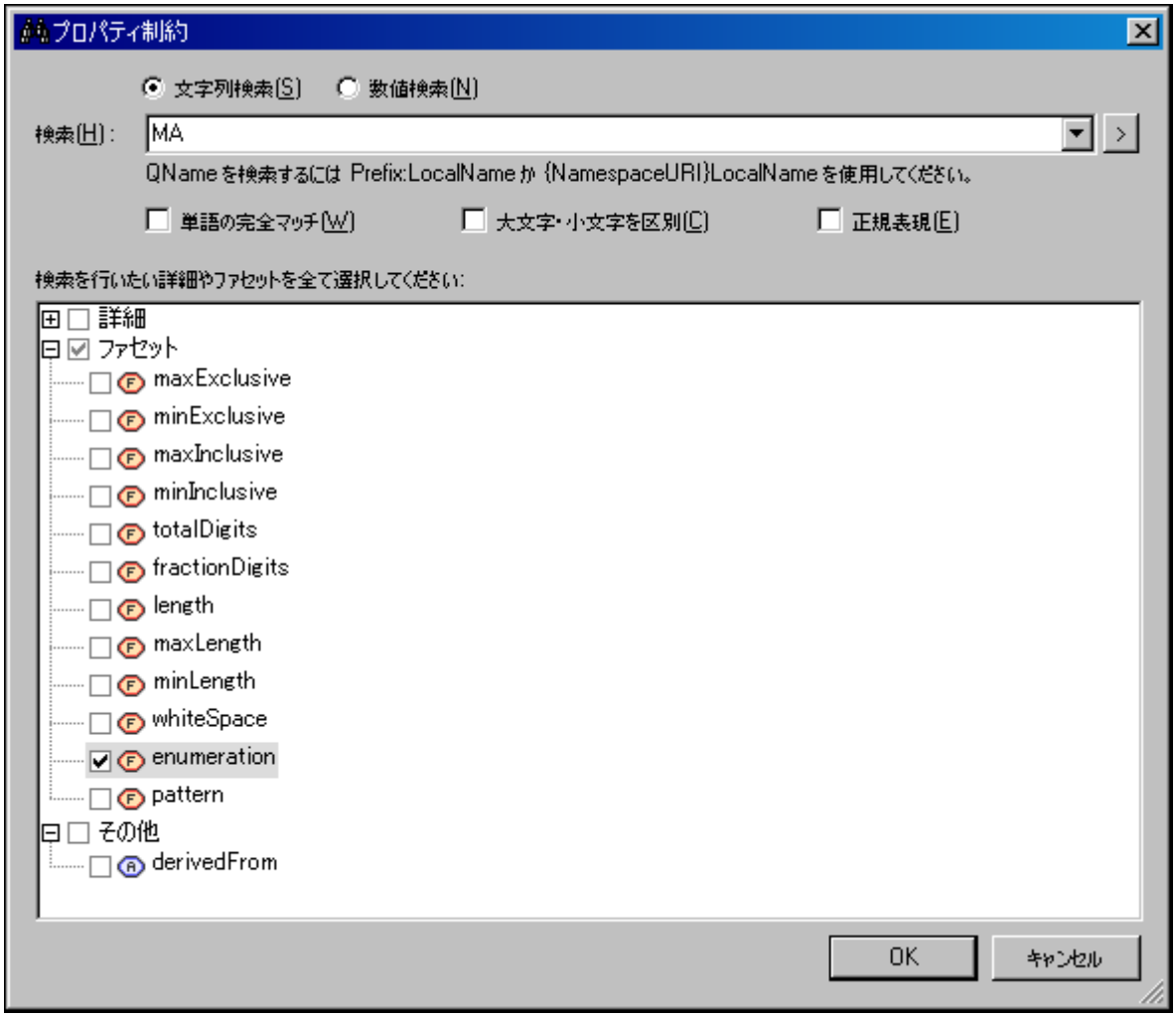
追加制約

追加制約により検索を行うプロパティの持つ値を指定することができます。例えば、列挙の MA（米国のマサチューセッツ州）という値を持つ state という名前の要素を探している場合、enumeration と MA という値を追加制約オプションで指定することで希望する検索を行うことができます。以下のような操作を行ないます:

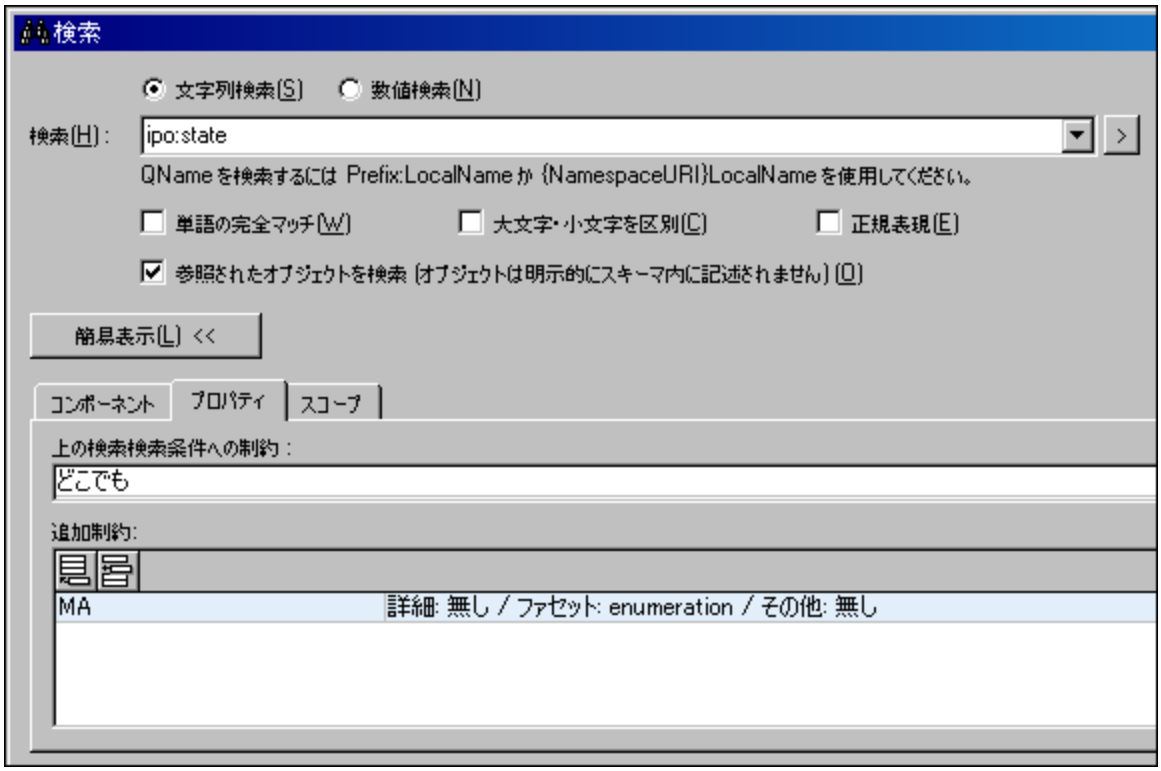
1. 追加制約ペインにて「**追加**」または「**挿入**」アイコンをクリックします（*下のスクリーンショット*）。



2. ペインに新たな行が追加され、プロパティ制約ダイアログが表示されます。全てのアイテムを選択解除して、enumeration プロパティのみを洗濯します（*下のスクリーンショット*）。



3. ダイアログの上部にあるテキストフィールドに、検索を行う列挙の値（この例では MA を入力します（*上のスクリーンショットを参照してください*）。
4. 「OK」をクリックします。（*以下のスクリーンショットにあるように*）追加制約ペインにて新たに作成された行に制約が入力されます。



上で示されるスクリーンショットにて検索語が `ipo:state` となっていることに注目してください。プロパティタブにある「どこでも」により、全てのプロパティが検索の対象となりますが、追加制約により検索は `MA` という値を持つ列挙に制限されます。

複数の追加制約により、検索結果を更に狭めることができます。追加制約を削除するには、追加制約を選択して**削除**アイコンをクリックしてください。

注意: プロパティタブまたは検索、置き換えダイアログが再度開かれたとしても、前回の追加制約選択状態は保持されたままになります。

6.6.4 スコープ

検索の範囲を(以下のスクリーンショットにある)スコープタブにより設定することができます。検索範囲をファイル、またはスキーマビューで現在選択されているスキーマコンポーネントから選択することができます。

ファイルオプションが選択されている場合、更に以下のオプションから選択範囲の指定を行うことができます:

- **現在のファイル**：現在アクティブなファイルが検索範囲になり、インクルード、インポート、再定義されたファイルも検索対象に含めるオプションが表示されます。
- 開いたファイル：XMLSpy で開かれている全てのXML スキーマ（XSD）ファイルが検索範囲になります。全て検索ならびに全て置き換えコマンドのみが有効になり、通常の検索を使用することはできません。
- **プロジェクト**：現在アクティブなプロジェクトが検索範囲になり、外部フォルダーをスキップするオプションが表示されます。全て検索ならびに全て置き換えコマンドのみが有効になり、通常の検索を使用することはできません。（「**ツール | オプション | ファイルの種類 | デフォルトビュー**」で設定することができる）`.xsd` ファイル拡張子のデフォルトビューがスキーマビューでない場合、`.xsd` ファイルは検索対象となりません。
- **フォルダー**：検索範囲となるフォルダーを指定することができ、サブフォルダーを含めるオプションが表示されます。全て検索ならびに全て置き換えコマンドのみが有効になり、通常の検索を使用することはできません。（「**ツール | オプション | ファイルの種類 | デフォルトビュー**」で設定することができる）`.xsd` ファイル拡張子のデフォルトビューがスキーマビューでない場合、`.xsd` ファイルは検索対象となりません。
- インクルード/インポート/再定義されたファイルのオプションをチェックすることで、これらのファイルを検索範囲に含めることができます。

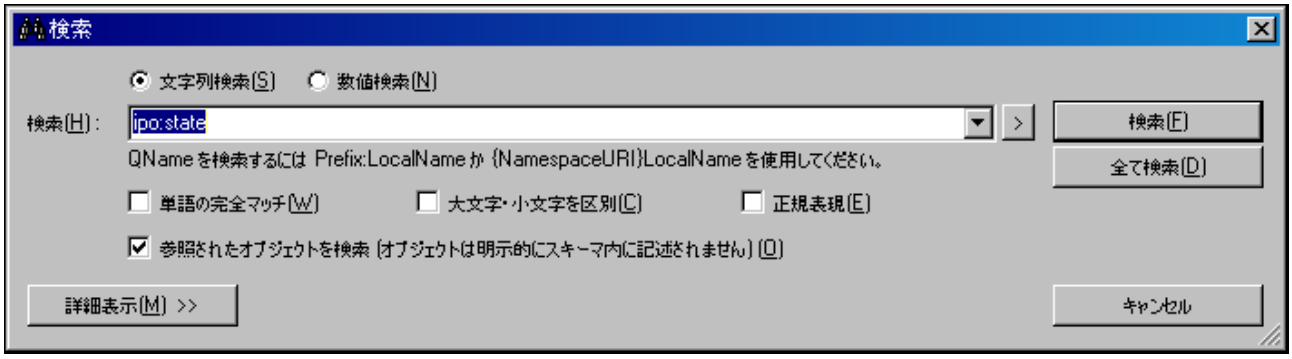
置き換えダイアログでは、ディスク上のファイルを直接置き換えるか、ファイルを XMLSpy にて開くかを選択することができます。ダイアログにあるボタンから選択してください。

6.6.5　検索と置き換えコマンド

検索ならびに置き換えダイアログ内の「**検索**」コマンドの動作はこれまでの検索方法とは多少異なっています。ここでは両ダイアログの「**検索**」コマンドならびに「**置き換え**」コマンドの解説を行ないます。

検索ダイアログ

検索語、そして必要な場合はその他の条件を入力した後、「検索（F3）」または「全て検索」コマンドをクリックします（*下のスクリーンショット*）。



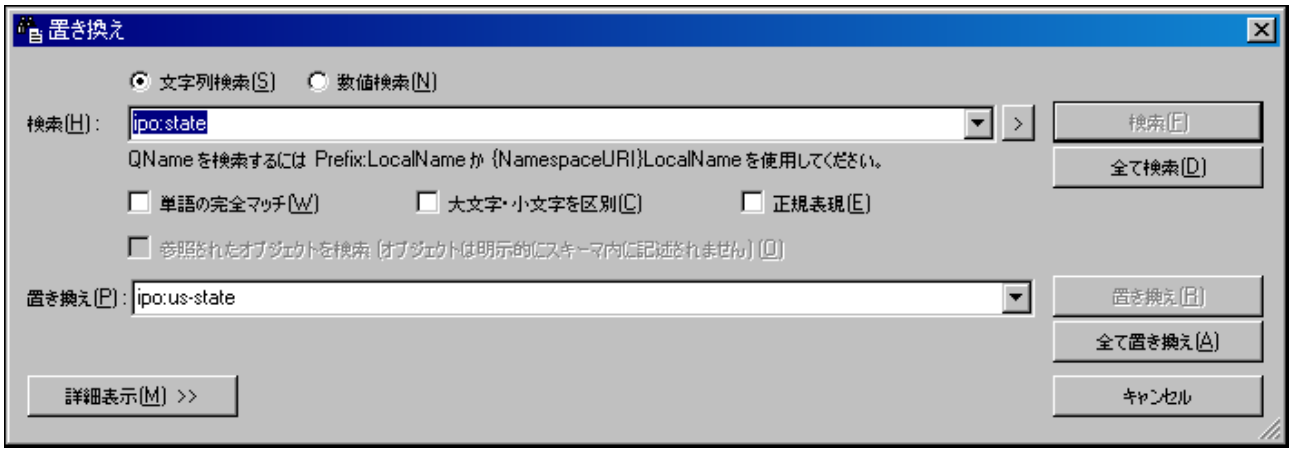
ダイアログにある「**検索** (Ctrl+F)」コマンドをクリックすることにより検索ダイアログが閉じられ、(検索条件や検索語といった) 検索条件にマッチする検索範囲内の次の出現場所が特定されます。スキーマビューで現在選択されているコンポーネントの位置を基準にして、次の検索結果は求められます。検索範囲の最後まで検索が行われても、検索範囲の最初から自動的に検索が続けられることはありません。そのため、検索を始める前に、現在選択されているコンポーネントの位置が検索を行ないたい箇所の前にあることを確認する必要があります。

「**検索**」の結果はスキーマビューにてハイライトされ、スキーマの中を検索ウィンドウにもその結果が表示されます。スキーマの中を検索ウィンドウにある結果をクリックして、スキーマビューの対応するアイテムをハイライトさせることもできます。

「**全て検索**」コマンドをクリックすると、検索ダイアログが閉じられ、検索結果の全てがスキーマの中を検索ウィンドウに表示されます。

置き換えダイアログ

(*下のスクリーンショット*) 置き換えダイアログでは「**検索**」コマンドをクリックすることで、スキーマビューで現在選択されている箇所から、次の検索語の出現箇所を特定します。「**置き換え**」をクリックすることで、この出現を置き換えることができます。



「**全て検索**」コマンドを使うと、置き換えダイアログが閉じられ、検索の結果全てがスキーマの内を検索ウィンドウに表示されます。

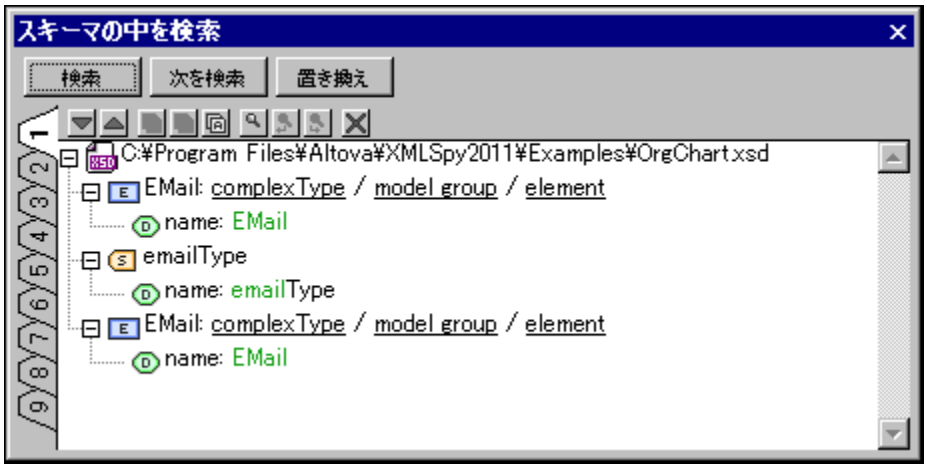
「**全て置き換え**」コマンドは、検索により見つかった箇所全ての置換えを行ないます。

メモ 置換フィールド内で正規表現はサポートされていません。

(C) 2015-2021 Altova GmbH

6.6.6 結果と情報

「検索」、「全て検索」、「置き換え」、または「全て置き換え」コマンドが実行されると、コマンド実行の結果が（以下のスクリーンショットにあるように）スキーマの中を検索ウィンドウに表示されます。検索された単語は緑色で表示されます（以下のスクリーンショットの例だと検索語の `email` が緑色で表示されます）。スキーマファイル内の位置も表示されていることに注目してください。



「全て検索」または「全て置き換え」コマンドを使うと、ドキュメント内の出現箇所全てが表示されます。

メモ: ウィンドウの上部にある検索や置き換えボタンを使えば、それぞれ検索、置き換えダイアログを表示させることができます。「次を検索」ボタンを使えば、検索語が次に出現する箇所を特定することができます。

スキーマの内を検索ウィンドウの機能

検索などの結果は（1 から 9 までの番号が振られた）9 つの異なるタブの中に表示されます。例えばある検索結果を 1 つのタブに収めておき、別のタブで新たな検索をすることにより結果の比較を行うことができます。スキーマの中を検索ウィンドウの結果をクリックすることで、メインウィンドウのスキーマビューに表示されている関連するコンポーネントがハイライトされます。これらのナビゲーション機能により、目的のコンポーネントへ素早くたどり着くことができます。

スキーマの中を検索ウィンドウでは以下のツールバーコマンドを利用することができます:

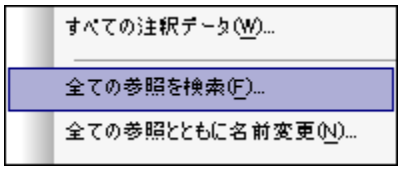
- 「次」ならびに「前」アイコンにより、現在選択されているメッセージの次（もしくは前）にあるメッセージが選択されます。
- 「メッセージをコピー」コマンドにより、選択されたメッセージ、子を含む選択されたメッセージ、そして全てのメッセージをクリップボードにコピーします。
- 「検索」コマンドによりスキーマの中を検索ウィンドウ内にあるテキストを検索します。
- 「クリア」コマンドにより、現在アクティブなタブのメッセージが全て消去されます。

6.6.7 グローバルコンポーネントの検索と名前変更

ユーザーは、選択されたファイルの XML スキーマや、関連する全てのスキーマファイル内にある名前付きグローバルコンポーネントの検索ならびに名前変更を行うことができます。名前を持つグローバルコンポーネントは以下を除いてすべてグローバルコンポーネントです: Include, Import, Redefine, Annotation, Comment, and PI components

このプロセスの仕組みは以下のとおりです:

1. スキーマ概要にて検索または名前の変更を行うグローバルコンポーネントを選択します。
2. 選択されたコンポーネントを右クリックすることで表示されるコンテキストメニューから、目的のコマンド（「**全ての参照を検索**」または「**全ての参照とともに名前変更**」）を選択します。

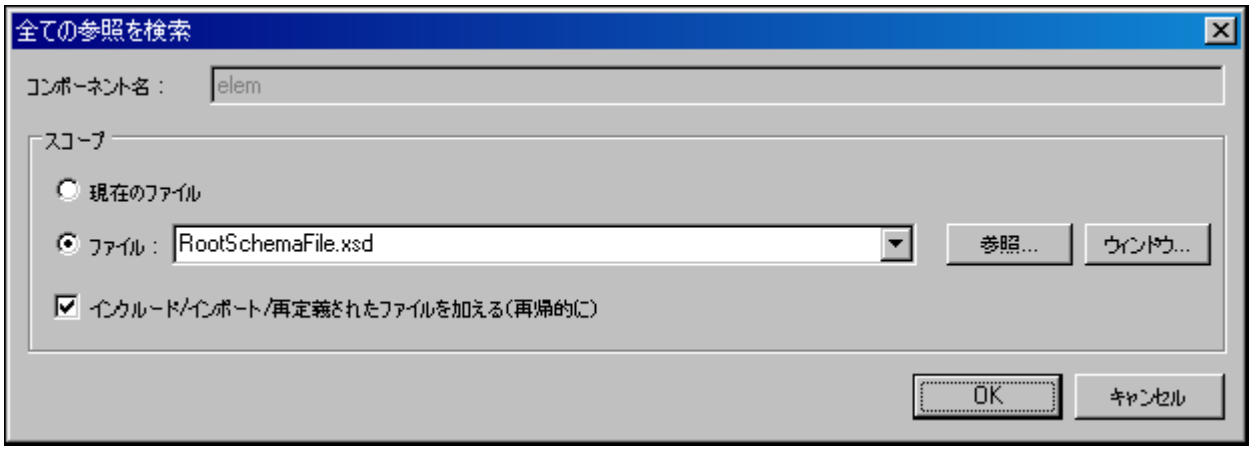


3. 表示されるダイアログにて、検索を行う範囲（または名前変更オペレーション）を選択します。名前変更オペレーションの場合、新たなグローバルコンポーネントの名前を入力します。
4. 「OK」をクリックすると検索の結果はスキーマウィンドウ内の検索内に表示されます（*下のスクリーンショットを参照してください*）。

グローバルコンポーネントへの参照を行っているファイルの場所が表示されます（*上のスクリーンショットを参照*）。名前の変更が行われたコンポーネントも全て表示されます。

全ての参照を検索

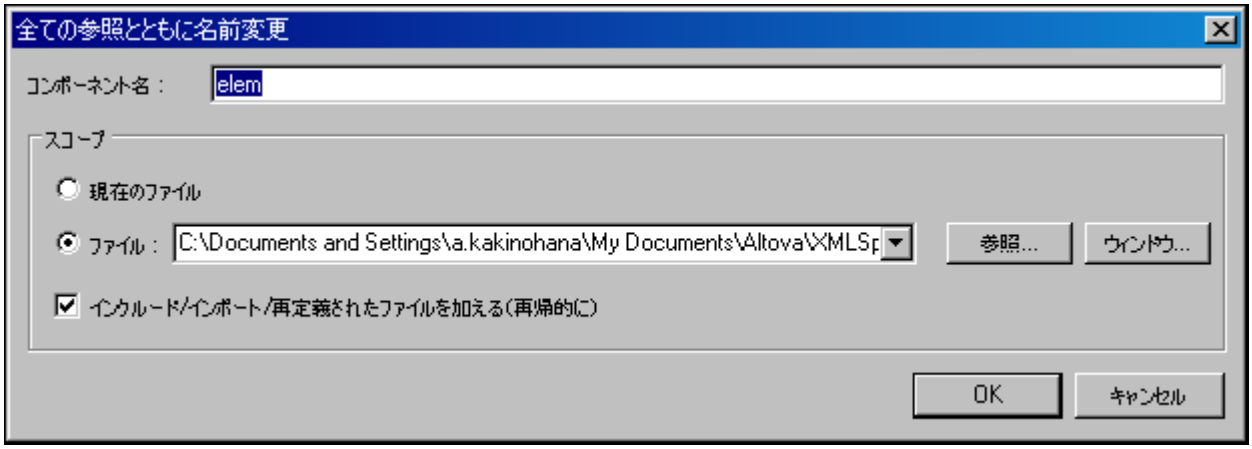
全ての参照を検索ダイアログ（*下のスクリーンショット*）を開くには以下を行います: **全ての参照を検索**ダイアログ（以下のスクリーンショットを参照）を開くには、(i) スキーマ概要にてグローバルコンポーネントを右クリックし、(ii) コンテキストメニューから、「全ての参照を検索」コマンドを選択します。



グローバルコンポーネント名が*コンポーネント名*フィールドにグレーで表示されます（編集は行えません）。検索の範囲対象を現在のファイルで行うか、または参照により選択したファイル内で行うかを指定することができます。ダイアログ下部にある*参照されているスキーマファイルの追加*チェックボックスを有効にすることで、（Include、Import、Redefine により）関連するファイルを検索の対象に加えることもできます。

全ての参照とともに名前変更

グローバルコンポーネントの名前を変更するには、コンポーネントを右クリックし、表示されるコンテキストメニューから「**全ての参照とともに名前変更**」を選択します。全ての参照とともに名前変更ダイアログが表示されます（*以下のスクリーンショットを参照*）。



選択されたグローバルコンポーネントに対して、新たに使用したい名前を*コンポーネント名*テキストフィールドに入力します。検索ならびに名前変更の範囲対象を、現在のファイルにするか、または参照により選択されたファイル内で行うかを指定することができます。ダイアログ下部にある*参照されているスキーマファイルの追加*チェックボックスを有効にすることで、（Include、Import、Redefine により）関連するファイルを検索の対象に加えることもできます。

7 XSLT

Altova ウェブサイト : XSLT エディター

このセクションでは以下のような構成でXSLT の説明を行ないます:

- XSLTドキュメントの編集 321 : XMLSpy におけるXSLTドキュメントの編集機能について説明します。
- XSLT 処理 323 : 選択されたエンジンを使って XMLSpy GUI からXSLT 変換を行う方法について解説します。このセクションではXMLSpy で行うことができるXSLT の設定についても説明されます。
- XSL アウトライン 325 : 複数のXSL ファイルの閲覧、ナビゲーション、そして管理を行うのに便利な XSL アウトラインならびに XSL 情報ウィンドウについて説明します。

XPath の評価

XML ドキュメントがアクティブになっている際、XPathウィンドウ 108 を使用することで、XPath 条件式の評価を行うことができます。この機能により、XPath 条件式が評価される様子を素早く確認することができます。XPath 条件式を入力すると、ドキュメントルートまたはXMLドキュメント内で選択されたコンテキストノードに相対的な評価が行われます。条件式評価の結果は、すぐにXPath ウィンドウにて表示されます。XPath ウィンドウの使用方法については、イントロダクション | XPath ウィンドウ 108 のセクションを参照ください。

XSLT デバッガー

XMLSpy にはXSLT デバッガー 356 が搭載されており、正確で効率のよいXSLT スタイルシートを素早く作成することができます。これらの機能はXSLT とXQuery 320 セクションにて記述されます。

その他のXSLT 機能

このセクションで記述される様々な機能のより詳しい情報については、ユーザーリファレンスの関連するメニューコマンド 886 を参照ください。

Altova XSLT エンジン

Altova XSLT 1.0、2.0 ならびに3.0 エンジンについては付録 1481 のXSLT とXQuery エンジンの情報 1482 を参照ください。

RaptorXML を使ったコマンドラインやバッチ処理

XMLSpy GUI のプロジェクト機能を使用して、バッチ処理を行うことができます。より柔軟性のある処理を求める場合は、速いXML 検証 XSLT 変換、XQuery実行機能を搭載したAltova のRaptorXML 製品を使用してください。をお使いください。コマンドラインやバッチ処理によりXSLT 変換を行うのにRaptorXML は最適のツールです。

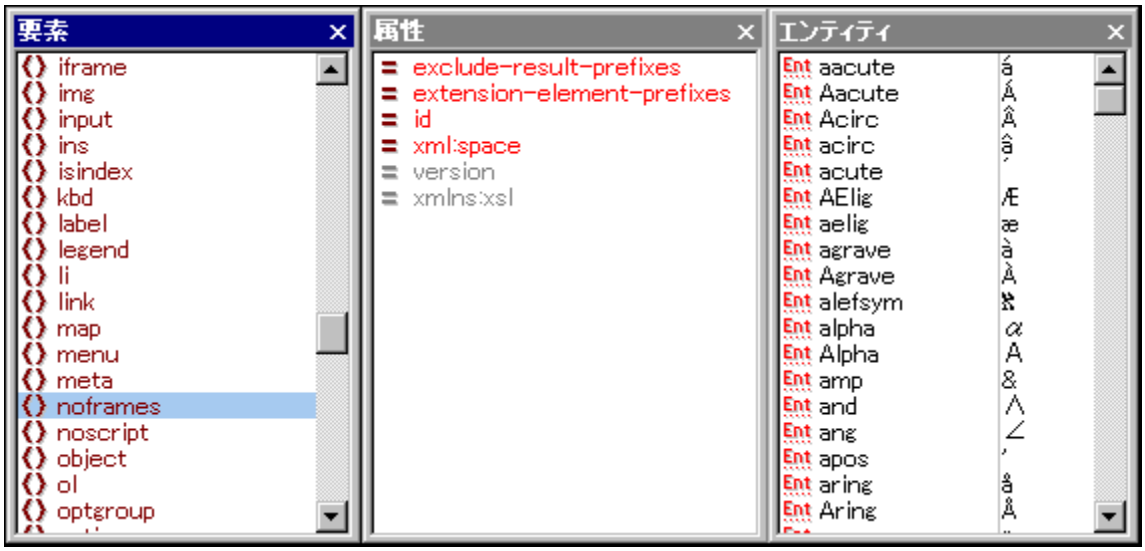
(C) 2015-2021 Altova GmbH

7.1 XSLT ドキュメント

XSLT 1.0、2.0、および3.0 ドキュメントは、その他のXML ドキュメントと同じようにテキストビュー122 やグリッドビュー141 にて編集することができます。ドキュメントが開かれるデフォルトのビューは、オプションダイアログのファイル型のセクションにて設定することができます。

入力ヘルパー

要素、属性、そしてエンティティ入力ヘルパーを利用することができます。入力ヘルパーにて表示されるアイテム情報はXMLSpy に内蔵されており、XSLT ドキュメント内で行われる参照とは独立したものです。



以下の点に注意してください:

1. 新規 XSLT ドキュメントが「**ファイル | 新規作成**」メニューの新規ドキュメント作成ダイアログにて作成されたとき、(XSLT 1.0、XSLT 2.0、またはXSLT 3.0 により作成されたドキュメントの種類によって違いはありますが)適切な XSLT 要素と属性が入力ヘルパーにロードされます。それに加えてHTML 要素や属性、HTML 2.0 エンティティセット、Latin-1、特殊な文字、そしてシンボルがロードされます。
2. 「**ファイル | 新規作成**」の新規ドキュメントダイアログにてXML ドキュメントが作成され、XSLT コンテンツがそこから作成された場合、XML 文字エンティティ以外のアイテムは入力ヘルパーに表示されません。
3. 新規ドキュメントダイアログを通して作成されたXSLT ドキュメントが開かれた場合、上の1番目の様に、入力ヘルパーにアイテムが表示されます。
4. 新規ドキュメントダイアログにてXSLT ドキュメントとして作成されなかったXSLT ドキュメントが開かれた場合、上の1番目の様に入力ヘルパーにXSLT アイテムが表示されます。更に、XSL-FO 要素や属性がテキストビュー入力ヘルパーに表示されます。
5. 要素入力ヘルパーにある要素のプレフィックスは以下の様に固定されています: XSLT 要素の`xsl:`、HTML 要素のプレフィックス無し、XSL-FO 要素の`fo:`。入力ヘルパーを使用するには、入力ヘルパーに表示される内蔵のプレフィックスに合致させる形で、XSLT ドキュメントで宣言された名前空間にもプレフィックスが記述されている必要があります。

自動補完

テキストビューでは入力中に自動補完のポップアップが表示され、入力されたテキストにマッチした最初のアイテムがハイライトされます。要素が入力されている時、要素のリストがポップアップされ、アルファベット順で最も近いアイテムがハイライトされます。同様に、属性が入力されているとき、適用することのできる属性のリストがポップアップで表示されます。入力ヘルパーの以前のセクションで記述されたルールに従ってリストに表示されるアイテムは決定されます。

XPath インテリジェント編集

XSLT ドキュメントにて XPath 条件式を入力することができる場合（例えば、select 属性の値など）、インテリジェントな XPath 編集機能を利用することができます。

- 入力中にかっこを一致させるなどの、XPath のための構文の色分けをビルドします。
- XPath 関数にカーソルをポイントすると、関数に関する情報が含まれるヒントが表示されます。
- XPath 関数と軸は、入力中にポップアップ内で提示されます。上/下 カーソルを使用して、提示されたリストを上下に移動することができます。ポップアップ内で関数がハイライトされる場合、関数（署名）に関する情報が追加ポップアップ内に表示されます。
- XML ファイルが情報ウィンドウ329に割り当てられている場合、XML ファイルの要素ならびに属性がポップアップに表示されます。

XSLT ドキュメントの検証

XMLSpy に内蔵されている XSLT スキーマに対して XSLT ドキュメントの検証を行うことができます（「XML | XML 検証(F8)」をクリックします）。XSLT ドキュメントが XSLT 1.0、 2.0 および 3.0 （`xsl:stylesheet` 要素の `version` 属性により指定されます）により、適切な内蔵スキーマが選択されます。

(C) 2015-2021 Altova GmbH

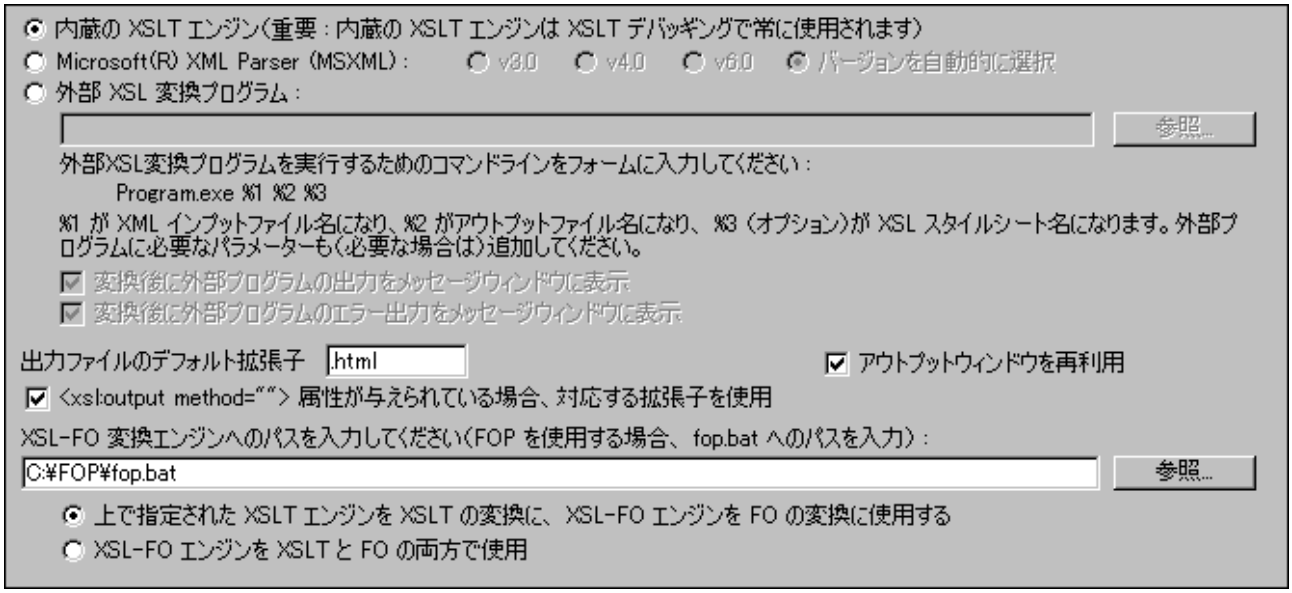
7.2 XSLT 処理

XMLSpy GUI では、2種類のXSLT 変換を利用することができます:

- 「XSL/XQUery | XSL 変換(F10)」コマンドにより、XSLT スタイルシートを使ったXML 変換処理を行ない、スタイルシートに記述されたフォーマットに従った変換が行われます。
- 「XSL/XQuery | XSL:FO 変換」コマンドにより、(i) XML からFO、そしてFO からPDF への2ステップの変換、または (ii) FO からPDF への変換が行われます。

変換を行うXSLT プロセッサーの指定

変換に使用されるXSLT エンジンはオプションダイアログのXSL タブ[1036]にて指定されます(以下のスクリーンショットを参照)。



利用可能なオプションについてはユーザーリファレンス[1036]セクションを参照ください。XSL セクションで指定されたエンジンがXSLT 変換で使用されます。XSL:FO 変換では、それに加えて他の(FO プロセッサーに同封されているXSLT という)XSLT エンジンオプションを使用することができます。このオプションを選択するには、XSL セクションの下部に表示される、対応するラジオボタンを選択してください。(*上のスクリーンショットを参照*)。

FO プロセッサーの指定

FO からPDF への変換時に使われるFO プロセッサーは、オプションダイアログのXSL タブ[1036]の下部にあるテキストフィールドから指定することができます(上のスクリーンショットを参照)。

XSLT 1.0、2.0、3.0 Altova のXSLT エンジン

スタイルシートのXSLT バージョンは`xsl:stylesheet` (または`xsl:transform`)要素の`version` 属性により指定されます。XMLSpy にはAltova XSLT 1.0、Altova XSLT 2.0、およびAltova XSLT 3.0 エンジンが内蔵されており、`version` 属性の値(`1.0` または`2.0` または`3.0`)により、適切なエンジンが使用されます。

XSLT 変換

XSL 変換 (F8) コマンドは以下のようなシナリオで使用することができます:

- GUI にてアクティブかつ XSLT ドキュメントが割り当てられている[894] XML ドキュメントの変換。XSLT ドキュメントが割り当てられていない場合、「XSL 変換(F8)」コマンドを選択した際に割り当てを行うよう求められます。
- GUI にてアクティブな XSLT ドキュメントの変換。「XSL 変換(F8)」コマンドをクリックすることで、アクティブな XSLT スタイルシートにより変換を行なう XML ファイルを選択するよう促されます。
- プロジェクトフォルダーやファイルの変換。プロジェクトフォルダーまたはファイルを右クリックし、コマンドを選択してください。

XSL:FO 変換

XSL:FO 変換コマンドは以下のようなシナリオで使用することができます:

- GUI にてアクティブかつ XSLT ドキュメントが割り当てられている[894] XML ドキュメントの変換。XML ドキュメントは、まず指定された XSLT エンジンを使って FO に変換されます。その後、FO ドキュメントは指定された FO プロセッサーにより PDF 出力へと変換されます。XSLT ドキュメントが割り当てられていない場合、「XSL:FO 変換」コマンドを選択した際に割り当てを行うよう求められます。
- 指定された FO プロセッサーにより FO ドキュメントを PDF へ変換。
- GUI にてアクティブな XSLT ドキュメントの変換。「XSL:FO 変換」コマンドをクリックすることで、アクティブな XSLT スタイルシートにより変換を行なう XML ファイルを選択するよう促されます。
- プロジェクトフォルダーやファイルの変換。プロジェクトフォルダーまたはファイルを右クリックし、コマンドを選択してください。

XSL:FO 出力ダイアログ[888]についての詳細は、ユーザーリファレンス[888]を参照ください。

XSLT のパラメーター

Altova XSLT エンジンを使用しているのであれば、XSLT パラメーターを GUI ダイアログに保管しておくことができます。全ての保管されたパラメーターは、変換時に XSLT ドキュメントへと送られます。詳しい情報については、XSLT パラメーター/XQuery 変数[890]コマンドに関する記述を参照ください。

RaptorXML を使ったバッチ処理

RaptorXML は無料のスタンドアロンアプリケーションで、Altova の最新 XML バリデーター、XSLT エンジン、そして XQuery エンジンが搭載されています。コマンドライン、COM インターフェース、Java プログラム、そして .NET アプリケーション内から使用することができ、XML ドキュメントの検証、XSLT スタイルシートを使用した XML ドキュメントの変換、そして XQuery ドキュメントの実行を行うことができます。

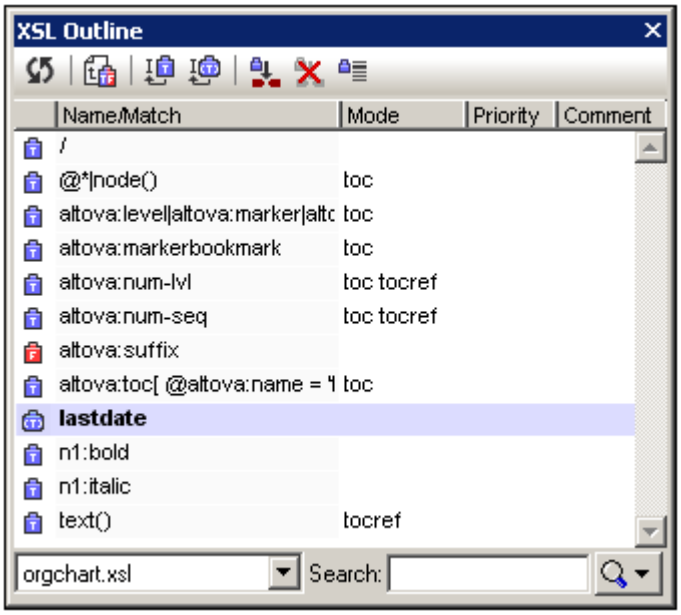
RaptorXML を使うと XSLT 変換タスクを自動化することができます。例えば、RaptorXML を呼び出してドキュメントのセットを変換するバッチファイルを作成することができます。詳細は RaptorXML ドキュメンテーションを参照ください。

(C) 2015-2021 Altova GmbH

7.3 XSL アウトライン

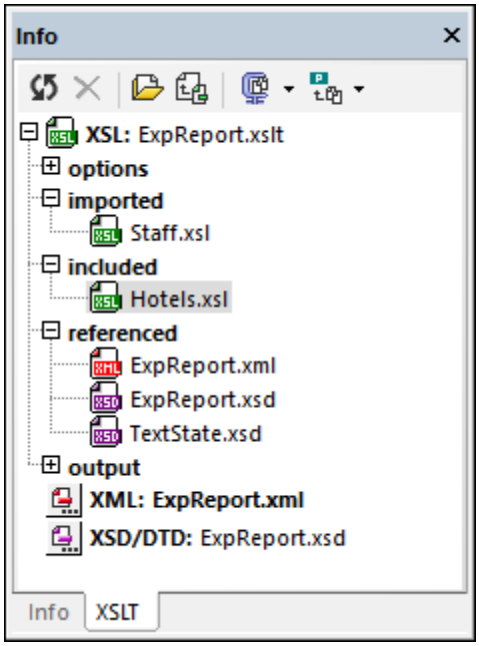
XSLT ドキュメントがXMLSpy のアクティブなドキュメントである場合、ドキュメントの構造に関する情報がXSL アウトラインウィンドウ326に表示され、XSLT ドキュメントに関連したファイル情報が(XSLT ドキュメントがXMLSpy でアクティブになっている時にだけ表示される)情報ウィンドウのXSLT タブ329に表示されます。更に、これら2つのウィンドウにて XSLT ドキュメントの編集や、関連するファイルの管理を行ないやすくする数々のコマンドが利用可能になります。

(下のスクリーンショットにある)XSL アウトラインウィンドウ326では、以下の操作を行うことができます:



- アクティブな XSLT ドキュメント、インポート/インクルードされた全てのドキュメントにあるテンプレートと関数を閲覧。
- 名前、マッチ、モード、優先度、またはコメントによるテンプレートと関数の並び替え。
- 名前/表現により特定のテンプレートを検索。
- XSL アウトラインを使ったXSLT ドキュメント内の対応するテンプレートをナビゲーション。
- 名前付きテンプレートに呼出しを素早く追加
- 選択された名前付きテンプレートを変換のエントリポイントとしてセット

(下のスクリーンショットにある)情報ウィンドウのXSLT タブ329を使って以下の操作を行うことができます:

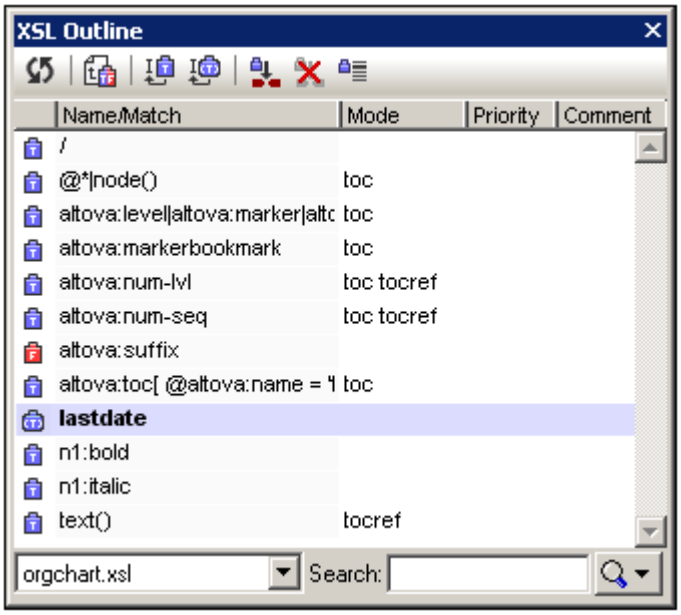


- インポート/インクルードされたファイルの位置といった、アクティブな XSLT ドキュメントに関連した全てのファイルに関する情報の閲覧。
- アクティブな XSLT ドキュメントを使って変換する XML ファイルのセット。更に、選択された XML ファイルの検証を行うスキーマ（XSD/DTD）ファイルをセットすることもできます。
- 情報ウィンドウから関連するファイルを開く。
- XMLSpy プロジェクトにて関連するファイルを素早く管理。
- 関連する全てのファイルを ZIP ファイルにまとめて、ユーザーの指定した場所に保存。

XSL アウトラインウィンドウ 326 ならびに情報ウィンドウの XSLT タブ 329 に関する詳しい情報は、このセクション以下のサブセクションを参照ください。

7.3.1 XSL アウトラインウィンドウ

（以下のスクリーンショットにある）XSL アウトラインウィンドウでは、アクティブな XSLT ドキュメントにある全てのテンプレートと関数が表示されます。テンプレートは青色のアイコン（パラメーターを持たないテンプレートは で、パラメーターを含むテンプレートは）で表示されます。関数は赤の アイコンで表示されます。ウィンドウの左下にあるコンボボックスを使うことで、(i)（以下のスクリーンショットにあるように）アクティブな XSLT ドキュメントからのみ、または(ii) アクティブな XSLT ドキュメントとインポート/インクルードされた全てのスタイルシートから得られるテンプレートと関数を表示するよう指定することができます。



テンプレートには、(i) 名前付きテンプレートと、(ii) XPath 条件式にマッチするテンプレートの2つの種類があります。各テンプレートは以下の情報と共に表示されます:

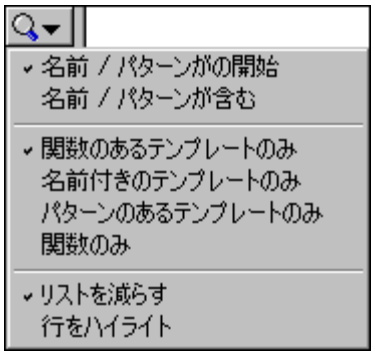
- (テンプレートに`name` 属性があれば) テンプレートの名前、そして(テンプレートに`match` 属性があれば) XPath 条件式。テンプレートが両方の属性を持っている場合、両方の情報が表示され、`name` 属性、`namevalue` 属性、そして`matchvalue` 属性の値と共に表示されます(スクリーンショットに太字で表示されているテンプレートに注目してください)。
- (モードがある場合) モード。テンプレートには1つ以上のモードがある可能性があることに注意してください。(*上のスクリーンショット参照*)。
- (優先度がある場合) 優先度。
- (テンプレートまたは関数に先行するコメントがある場合) コメント

スタイルシートの関数は名前により表示されます。関数にはモードも優先度もありません。

オペレーション

XSL アウトラインウィンドウでは以下のオペレーションを実行することができます:

- *フィルタリング*: ウィンドウに表示されているリストを、以下のうちのどれかを表示するようにフィルタリングすることができます: (i) 全てのテンプレートと関数(XMLSpy が起動されるたびに適用されるデフォルト設定)、(ii) 名前付きテンプレートのみ、(iii) XPath 条件式テンプレートのみ、(iv) 関数のみ。目的のフィルタリングを選択するには、ウィンドウ右下に位置している検索ボックスをクリックして、表示されたリスト(メニューの第2グループ)から目的のフィルターを選択します。選択されたフィルターはただちに適用され、設定が変更されるかXMLSpy を終了するまで有効になります。



- *並べ替えと検索*：列のヘッダーをクリックすることで、各列をアルファベット順に並べ替えることができます。再度クリックすることで並べ替えの順序を反転させることができます。この操作で列の並べ替えを行った後、リスト内で任意のアイテムを選択し、並べ替えられた列にて単語を入力すると、その単語を含むリスト内のアイテムがハイライトされます。この機能を使えば、目的の名前、条件式、モード、または優先度を持つテンプレートへ素早くたどり着くことができます。
- *検索*：（ウィンドウの右下にある）検索テキストフィールドに、検索を行ないたい名前または XPath 条件式を入力することができます。入力中にも検索の結果は表示されます。（上のスクリーンショットにある）検索ボックスに表示されるリストから、以下のオプションを指定することができます：（i）名前または条件式が検索語を含む（または検索語から始まる）。検索語から始まるオプションは XMLSpy が起動されるたびにデフォルトオプションとして指定されます。（ii）検索結果をハイライトするか、それ以外を表示しないよう設定。表示を減らすオプションは XMLSpy が起動されるたびにデフォルトオプションとして指定されます。これらの選択は直ちに適用され、設定を変更するか、XMLSpy を終了するまで有効になります。
- *再ロード*：スタイルシートが変更された後、ウィンドウのツールバーにある「**同期**」アイコン により XSL アウトラインを更新することができます。
- *アイテムへ移動*：XSL アウトラインウィンドウにてテンプレートまたは関数が選択されたとき、ウィンドウのツールバーにある「**定義へ移動**」アイコン をクリックすることで、ドキュメント内にあるテンプレートまたは関数がデザインビューでハイライトされます。もしくはエントリーをダブルクリックすることでも同様の効果を得ることができます。
- *名前付きテンプレートアクション*：名前付きテンプレートに関しては、2種類の操作を行うことができます。（i）（`xsl:call-template` を使った）名前付きテンプレートへの呼び出しをスタイルシート内のカーソル位置に挿入することができます。（ii）名前付きテンプレートを変換のエントリポイントとしてセットすることができます。これらのオペレーションのコマンドは以下に記述されるようにツールバー上のアイコンから起動することができます。

変換のためのテンプレートモード

変換のための設定モードと呼ばれるツールバーのコンボボックスには、以下がリストされています（i）スタイルシートの全てのモード（ii）XSLT 3.0 スタイルシートの場合（デフォルトモードを選択する）空のエントリ（iii）`#unnamed` モード。ドロップダウンリストからモードを選択して、選択されたモードを変換のために設定します。（すべての XSLT バージョンの）`#unnamed` モードは、`mode` 属性を持たないすべてのテンプレートモードに適用することができます。

XSLT 1.0 と XSLT2.0 スタイルシートの場合、デフォルトのモードは `#unnamed` です。ですから、空のエントリを選択することは、（`#unnamed` モードで、結果すべてのテンプレートに `mode` 属性無しで適用される）デフォルトのモードを選択することになります。

XSLT 3.0 スタイルシートでは、トップレベルの `xslt` 要素は、変換のためのデフォルトモードを持つ `default-mode` 属性を持つことができます。変換のためのモード設定コンボボックスでは、空のエントリ（デフォルトモード）が選択されていて、`default-mode` 属性で指定されているモードが変換モードとして使用されます。`#unnamed` モードがコンボボックスで選択されている場合、変換はすべてのテンプレートに名前のないモードで適用されます。すなわち、テンプレートにはモード属性がありません。

メモ テンプレートにモードの値 `#all` を与え、全てのモードに適用することができます。

(C) 2015-2021 Altova GmbH

名前付きテンプレート

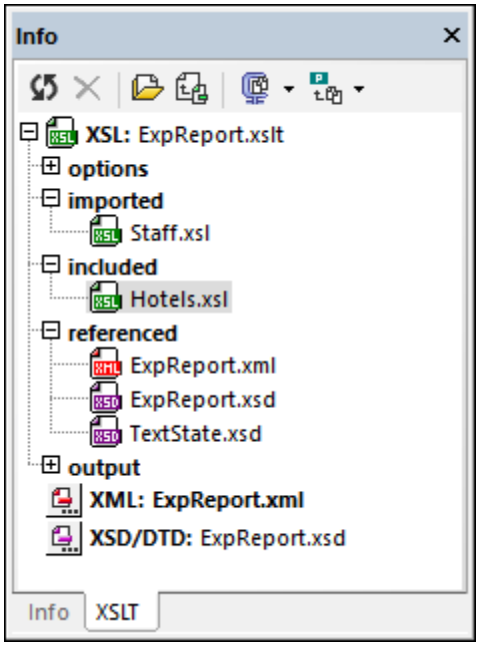
名前付きテンプレートが選択されていれば、（以下のスクリーンショットにあるように）ウィンドウのツールバーにある名前付きテンプレートに関連したコマンドが選択可能になります。

ツールバーのコマンドは左から順に以下のようになります:

- *xsl::call-template の挿入*：このコマンドはXSL アウトラインウィンドウにて名前付きテンプレートが選択された時に有効になります。このコマンドにより `xsl:call-template` 要素がスタイルシートのカーソル位置に挿入されます。スタイルシートに挿入される `xsl:call-template` 要素の `name` 属性値は、選択された名前付きテンプレートの `name` 属性の値になります。これにより `xsl:call-template` が選択された名前付きテンプレートへの呼び出しになります。
- *パラメーターを含む xsl:call-template の挿入*：このコマンドはXSL アウトラインウィンドウにて選択された名前付きテンプレートが `xsl:param` 要素を持っている時に選択可能になります。**xsl:call-template** の挿入コマンドのように、`xsl:call-template` 要素が挿入されますが、このコマンドの場合、選択された名前付きテンプレートの `xsl:param` 子要素に対する `xsl:with-param` 子要素も一緒に挿入されます。挿入される `xsl:call-template` と、その `xsl:with-param` 子要素の名前は、選択された名前付きテンプレートとその `xsl:param` 子要素の名前に対応します。
- *選択された名前付きテンプレートを変換のエントリポイントとしてセット*：名前付きテンプレートが変換のエントリポイントとしてセットされている場合、XMLSpy で実行される変換はこの名前付きテンプレートから開始されます。XSL アウトラインウィンドウでは、エントリポイントとなる名前付きテンプレートが（このセクションの始めにあるスクリーンショットにあるように）太字で表示されます。
- *名前付きテンプレートを変換のエントリポイントから解除*：名前付きテンプレートが変換のエントリポイントとしてセットされると選択可能になります。
- *変換のエントリポイントとしてセットされた名前付きテンプレートへ移動*：名前付きテンプレートが変換のエントリポイントとしてセットされると選択可能になります。変換のエントリポイントとしてセットされた名前付きテンプレートとは別のアイテムにXSL アウトラインウィンドウのフォーカスがある場合、このアイコンをクリックすることでXSL アウトラインウィンドウにある名前付きテンプレートがハイライトされ、素早くアクセスすることができるようになります。

7.3.2 情報ウィンドウ

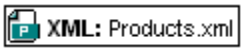
XMLSpy 内のアクティブなドキュメントがXSLT ドキュメントの場合のみ、情報ウィンドウのXSLT タブが表示されます。全てのインポートされたファイルおよびアクティブな XSLT ドキュメントに関連したXSLT ファイルを表示します。XSLT がアクティブドキュメントとして変換が開始された場合、XSLT を使用して変換するXML ファイルを選択することもできます。



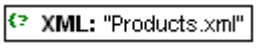
情報ウィンドウのXSLT タブには以下のファイルが表示されます:

- *XSLT ファイル*:(上のスクリーンショットにあるように)インポート/インクルードされた全てのXSLT ファイルが表示されます。ファイル上にマウスを移動させると、各ファイルの位置がポップアップで表示されます。インポートまたはインクルードされたファイルをダブルクリックするか、選択して 情報ウィンドウツールバーにある「開く」アイコンをクリックすることで、ファイルが新しいウィンドウで開かれます。ツールバーにある「Include/Import 位置へ移動」アイコンにより、アクティブな XSLT ドキュメントのinclude/import 宣言部がハイライトされます。

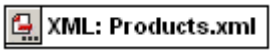
- *XML ファイル*:変換を行うためにXML ファイルをアクティブな XSLT スタイルシートに割り当てることができます。割り当てられたXML ファイルの場所は、ファイル上にマウスを移動すると表示されるポップアップで確認することができます。XML ファイルを指定した後、そのXML ファイル上でメニューコマンド「XML/XQuery | **XSL 変換**(F10)」を選択することで、定義済みのXSLT スタイルシートを使った変換が実行されます。XML ファイルは**XML** アイコンをクリックすることで選択することができます(選択されたファイルは太字で表示されます)。その他にもプロジェクトプロパティダイアログ(*XSL ファイルの XSL 変換*)やXSLT ドキュメントの処理命令(`<?altova_samplexml "products.xml"?>`)からXML ファイルを割り当てることができます。どのような方法でも、以下のようなアイコンと共にXML ファイルはInfo ウィンドウに表示されます:

  プロジェクトプロパティダイアログにより割り当てられている

  XSLT ドキュメントの処理命令により割り当てられている

  XML アイコンをクリックし、ファイルの参照を行うことにより割り当てられている(太字で表示されます)

  同じファイルに対して2つ以上の方法で割り当てがなされている場合、選択の優先順位は:(i) プロジェクト、(ii) 処理命令、(iii) ユーザーによる参照、となります。アイテムをダブルクリックするか、選択後に「開く」ツールバーアイコンをクリックすることでXML ファイルを開くことができます。

- *XSD/DTD ファイル*:選択されたXML ファイルがスキーマ(XML スキーマまたはDTD)への参照を含む場合、XSD/DTD エントリーに、そのスキーマファイルが表示されます。その他にも、(XML ファイルの場合と同じように)プロジェクトプロパティダイアログ(*検証*)からスキーマファイルを選択し、「XSD/DTD」アイコンをクリックし、目的のスキーマファイルを選択することもできます。スキ

ーマファイルがプロジェクトプロパティダイアログから選択された場合、エントリーの隣にプロジェクトアイコンが表示され、それ以外の場合には(「XSD/DTD」アイコンによりスキーマが選択された場合は太字で表示される)ファイルエントリーの隣に「XSD/DTD」アイコンが表示されます。スキーマファイルが2つ以上の方法で割り当てられている場合、選択の優先順位は:(i) プロジェクト、(ii) ユーザーによる参照、(iii) XML ドキュメント内の参照、となります。ファイルにマウスを移動させると、割り当てられた XSD ファイルの位置がポップアップで表示されます。ダブルクリックまたは選択後に「開く」ツールバーアイコンをクリックすることでスキーマファイルを開くことができます。

注意: XML または XSD/DTD ファイルがプロジェクトプロパティダイアログから選択された場合、この選択を解除するにはプロジェクトプロパティダイアログを開き、そこで選択を解除する必要があります。選択が「XML」または「XSD/DTD」アイコンにより選択がなされた場合、設定を解除するには、ファイルを選択して、情報ウィンドウツールバーにある「クリア」アイコンをクリックしてください。

オプション

XPath インテリジェント編集:XML ファイルが割り当てられている場合、XML ドキュメントの構造が走査され、インテリジェント XPath 編集 321 が要素と属性にも拡張されます。XSLT ドキュメント内の XPath 条件式を入力することができる場所では、使用可能な要素と属性がポップアップで表示されます。このオプションはデフォルトで有効になっており、チェックボックスのチェックを外すことで XPath インテリジェント編集を無効にすることができます。設定はファイルが閉じられるときに各 XSLT ファイルに個別に保存され、ファイルが開かれる際にその設定が読み込まれます。

ツールバーアイコン

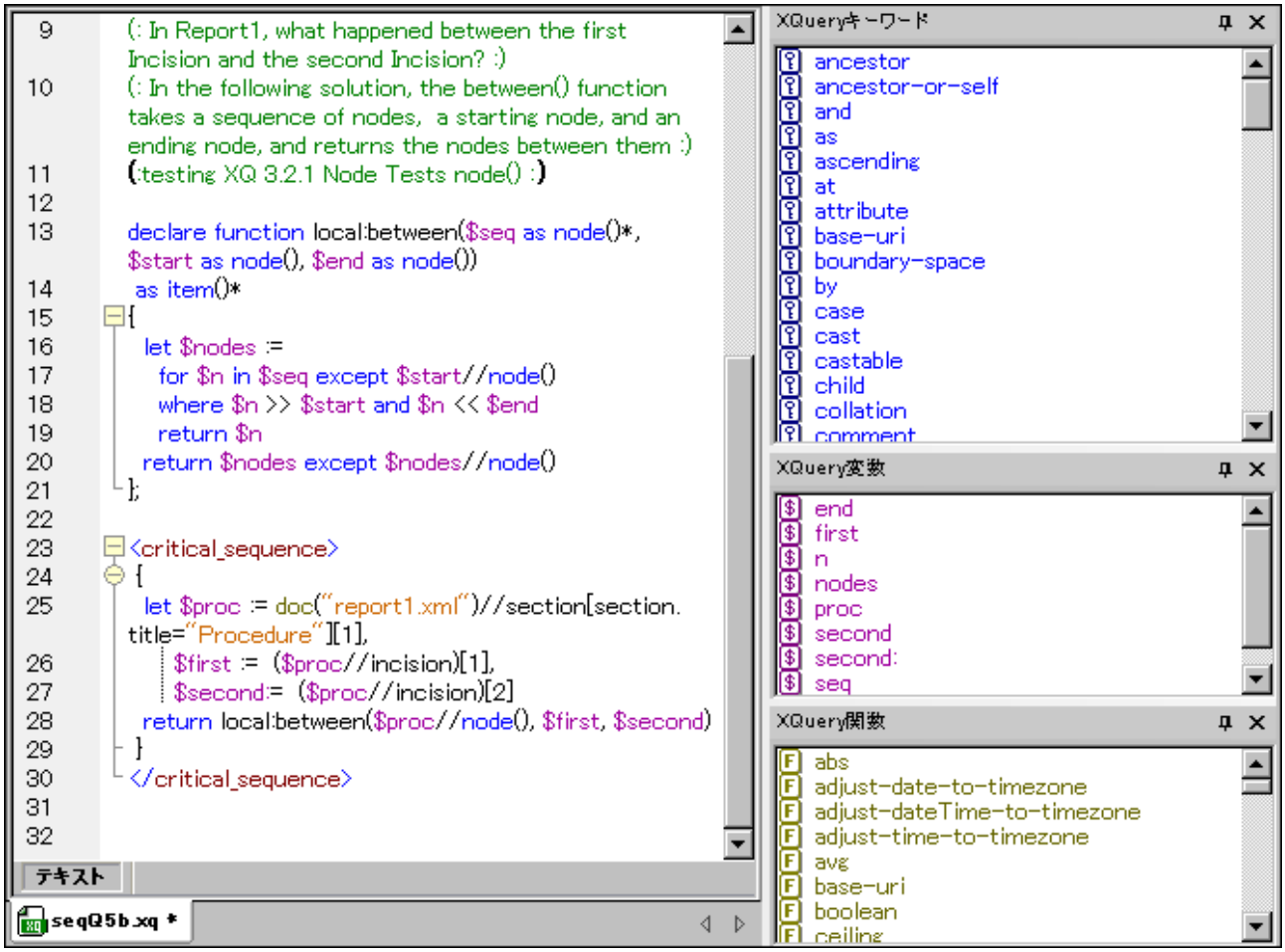
(以下のスクリーンショットにある)情報ウィンドウツールバーアイコンを、左から右に説明します:

- *再ロード*:XSLT ドキュメント内で行われた修正を反映するよう情報ウィンドウを更新します。
- *XML/XSD 割り当てをクリア*:XML または XSD/DTD アイコンを使ってなされた XML や XSD/DTD の割り当てを解除します。解除を行うファイルを選択して、このアイコンをクリックしてください。
- *ドキュメントを開く*:選択されたドキュメントを開きます。
- *import/include 位置へ移動*:インポートまたはインクルードされたファイルが選択されているとき、このアイコンをクリックすることで XSLT ドキュメント内の関連するインポートまたはインクルード宣言がハイライトされます。
- *全てのドキュメントを圧縮*:情報ウィンドウに表示されている全てのドキュメントを圧縮して、ユーザーの定義した場所に保存します。または、ドキュメントを選択して、アイコンのドロップダウンメニューから「**選択したドキュメントを圧縮**」コマンドを選択することで、選択されたファイルだけを圧縮することができます。
- *プロジェクトに全てのファイルを追加*:現在のプロジェクトに全てのファイルを追加します。または、ドキュメントを選択し、アイコンのドロップダウンメニューから「**プロジェクトに選択したファイルを追加**」コマンドを選択することで、選択したファイルだけをプロジェクトに追加することができます。

8 XQuery

Altova ウェブサイト：XQuery エディター

XQuery および XQuery Update 345 ドキュメントはテキストビューで編集することができます。このビューにより入力ヘルパーや、構文による色分け、インテリジェント編集といった機能を使用することができます。（*スクリーンショット参照*）また、XQuery の検証を行ない、XQuery ドキュメント内にあるコードを内蔵の Altova XQuery エンジンを使って実行することができます。



注意：XQuery ファイルはテキストビューでのみ編集することができます。その他のビューは XQuery に対してご利用になれません。

XQuery と XQuery Update ファイルの関連付け

XMLSpy 内では、XQuery と XQuery Update ドキュメントは 2 つの異なるドキュメントのタイプとして認識されます。通常 XQuery ドキュメントの拡張子は **.xq** で、XQuery Update ドキュメントのファイル拡張子は **.xqu** です。追加ファイル拡張子をこれらのファイルの型と関連付けることができ、ファイルの型の関連付けも、オプションダイアログのファイル型のセクション（ツール | オプション | ファイルの種類 1018）で変更することができます。

XQuery 実行または XQuery アップデート 345 により、「**XQuery/Update 実行**」コマンドが実行されるため、ファイル拡張子のドキュメントタイプの関連付けは重要です。

(C) 2015-2021 Altova GmbH

このセクション

このセクションは以下のように構成されています:

- XQuery ドキュメントの編集 334
- XQuery 評価 341
- XQuery の検証 342
- XQuery/Update 実行 343
- XQuery Update Facility 345
- XQuery とXML データベース 351

他の関連した機能と情報:

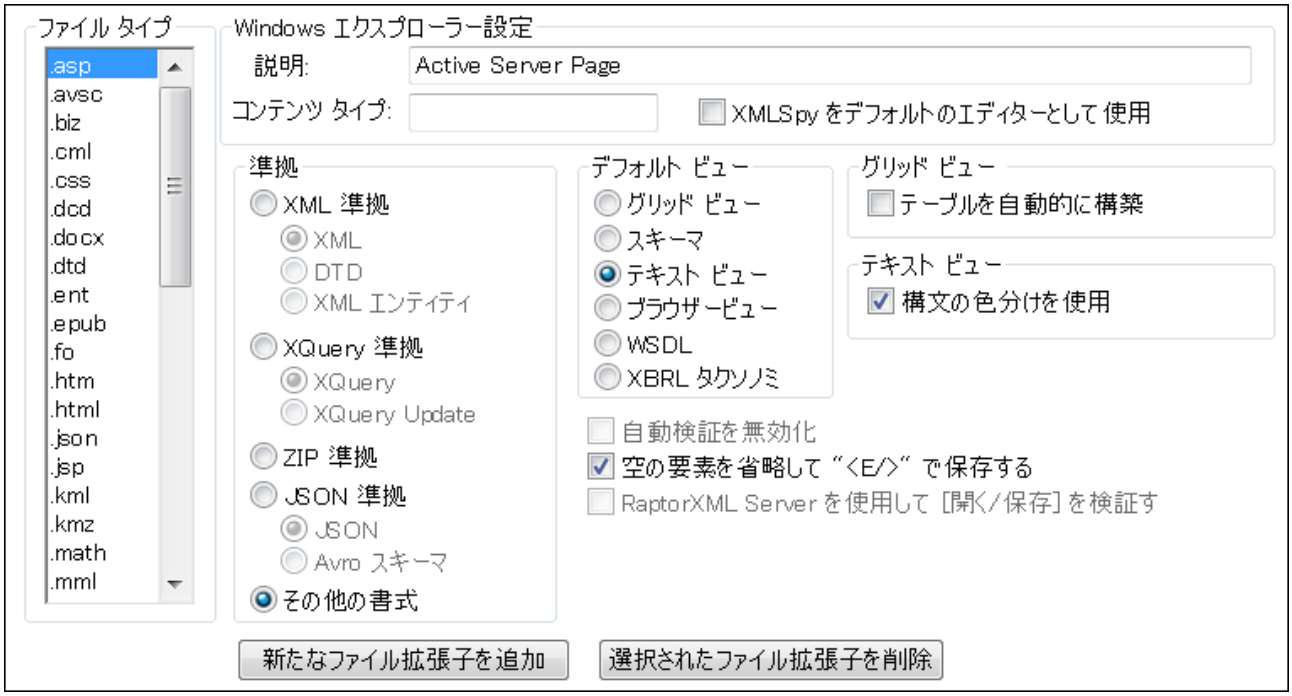
- XSLT/XQuery デバッガーとプロファイラー 355
- XQuery エンジンの実装 1482
- 出力ウィンドウ: XPath/XQuery 108
- ツール| オプション| ファイルの型 1018
- ツール| オプション| XQuery 1038

コマンドラインやバッチ処理のためのRaptorXML

XMLSpy のプロジェクト機能を使うことで、GUI からバッチ処理を行うことができます。より柔軟性のある処理をお望みの場合、XMLSpy にも搭載されているAltova XQuery エンジンを含むAltova のRaptorXML 製品を使用することができます。AltovaXML はコマンドラインやバッチ処理からXQuery の実行を行う理想的なツールになります。

8.1 XQuery ドキュメントの編集

XMLSpy 内では、XQuery とXQuery Update ドキュメントは2 つの異なるドキュメントのタイプとして認識されます。ドキュメントタイプ（XQuery またはXQuery Update）オプションダイアログのファイルの型内のファイル拡張子に割り当てられます。（ツール | オプション | ファイルの種類、1018 下のスクリーンショット）。XQuery またはXQuery Update タイプのファイルが、XMLSpy で開かれた場合、そのファイルのテキストビューのXQuery 編集機能を使用することができます。



XMLSpy でXQuery とXQuery Update として現在定義されているファイル拡張子

XQuery	`.xq` `.xql` `.xqr` `.xquery`
XQuery Update	`.xqu`

メモ このセクションで説明される編集機能はXQuery とXQuery Update ドキュメント内の説明と同様です。

XQuery /Update 実行

XSL/XQuery | XQuery/Update 実行 のGUI コマンドは、自動的に選択された、XQuery ファイルのXQuery 実行 またはXQuery /Update を実行します。詳細に関しては、「XQuery /Update 実行」343 を参照してください。

8.1.1 XQuery ドキュメント

XQuery またはXQuery Update ドキュメントはXQuery またはXQuery Update に準拠している場合、自動的に、テキストビューのXQuery 編集モードで開かれます。ファイルはオプションダイアログ（ツール | オプション | ファイルの型1018 下のスクリーンショット）のファイル型のセクションの特定のドキュメント型に準拠するように定義されています。

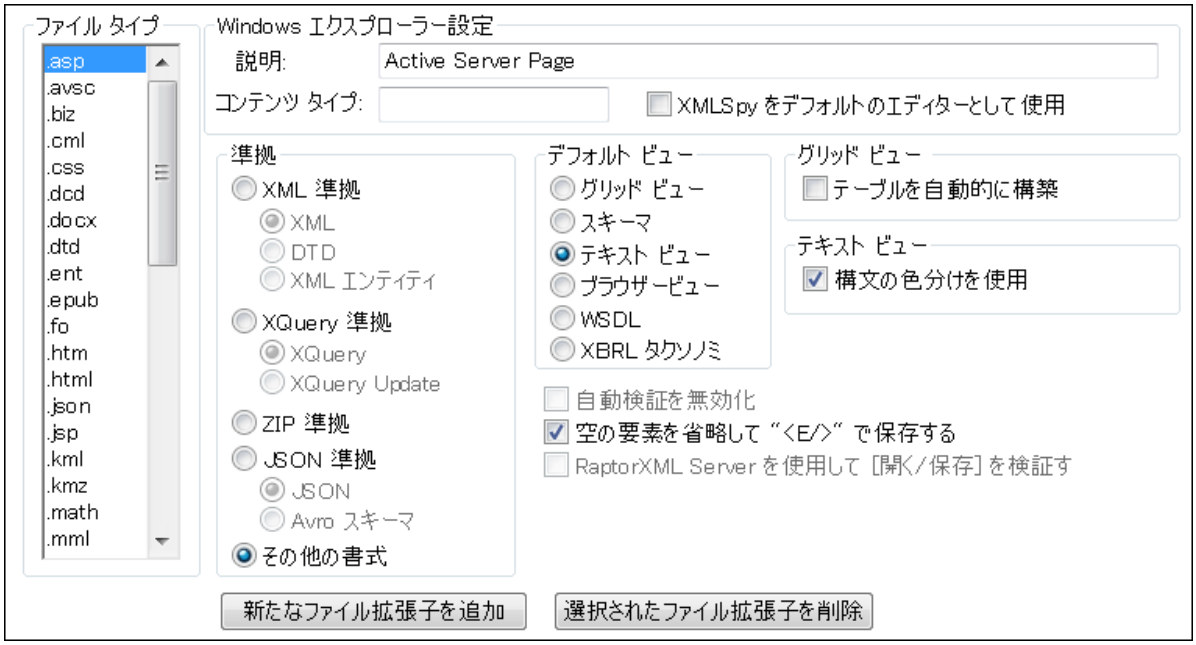
XMLSpy で XQuery と XQuery Update として現在定義されているファイル拡張子

XQuery	`.xq` `.xql` `.xqr` `.xquery`
XQuery Update	`.xqu`

XQuery 準拠のファイル拡張子を追加する

他のファイル拡張子を持つドキュメントを XQuery 準拠として認識させるには:

1. 「ツール | オプション」を選択。(以下のスクリーンショットにある) オプションダイアログが表示されます。



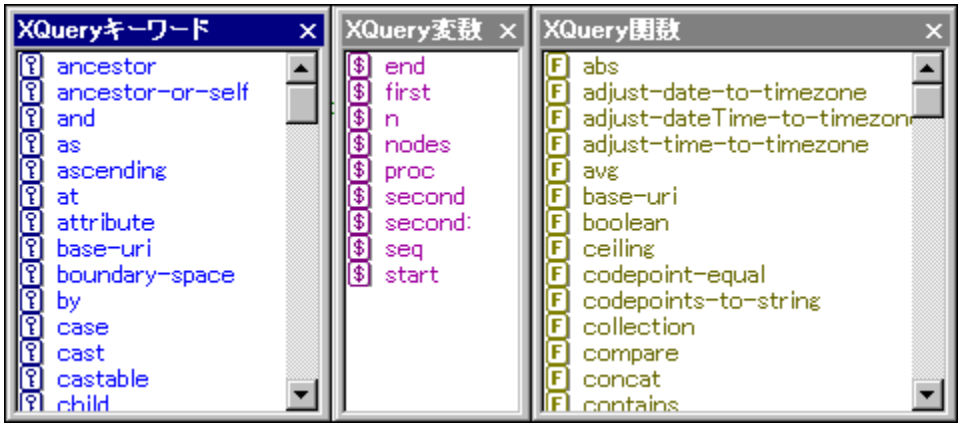
2. 「ファイルの型」セクションを選択。
3. 「新たなファイル拡張子を追加」をクリックして、ファイルの型のリストに新たな拡張子を追加。
4. 「準拠」にて、「XQuery 準拠」を選択。*XQuery* または *XQuery Update* を選択する。

次の設定も行ってください:

- *説明:* XML Query 言語 または XQuery Update Facility
- *コンテンツタイプ:* `text/xml`
- XMLSpy を XQuery ファイルの編集を行うデフォルトエディターとして使用する場合、「XMLSpy をデフォルトのエディターとして使用」チェックボックスのチェックを入れてください。

8.1.2　XQuery 入力ヘルパー

テキストビューの XQuery モードには3つの入力ヘルパーがあります: XQuery キーワード (青色)、XQuery 変数 (紫色)、XQuery 関数 (オリーブ色)。



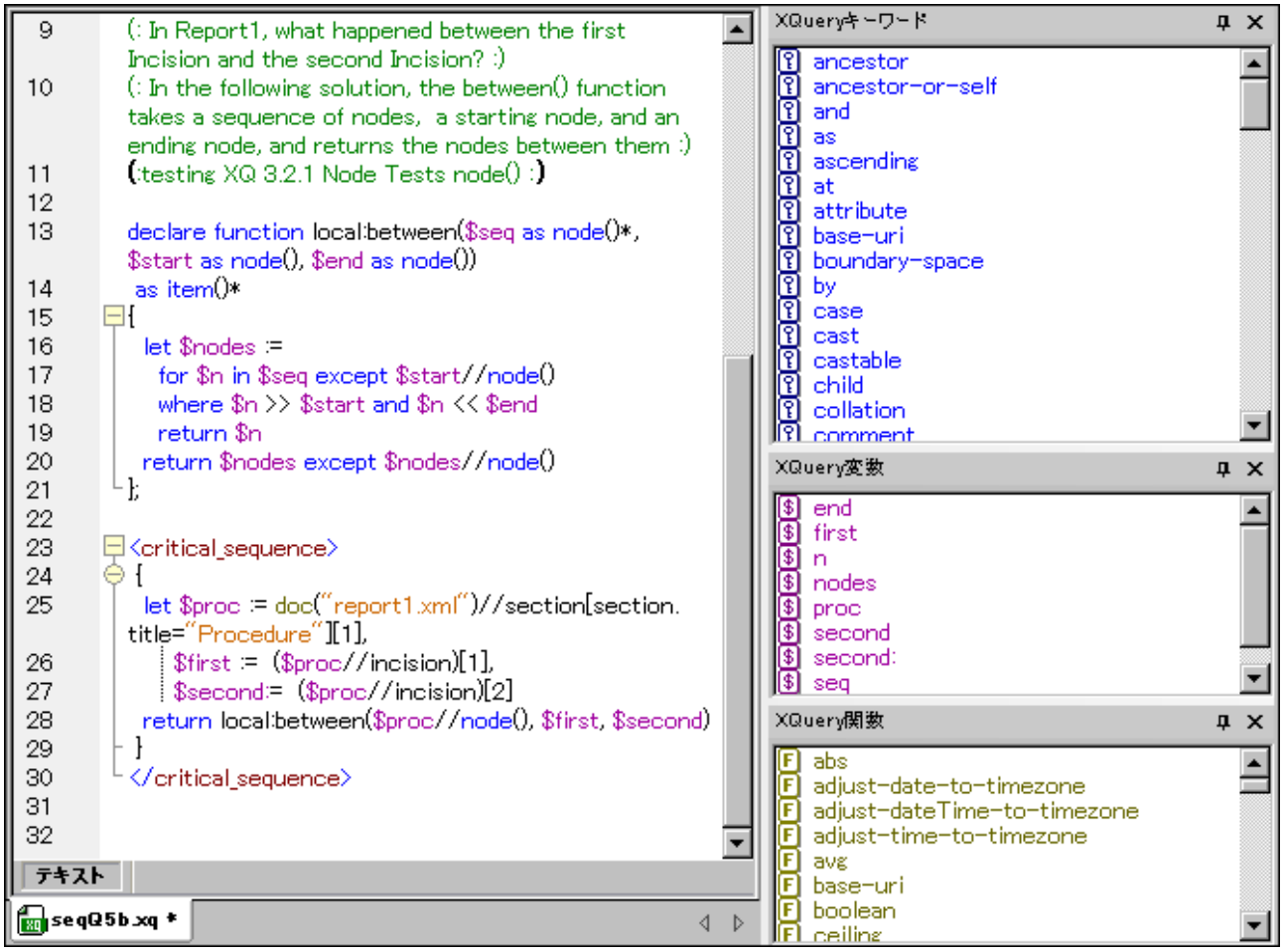
以下の点に注意してください:

- 入力ヘルパー内にあるアイテムの色は、それぞれテキストにて使用されている構文の色に対応しています。これらの色を変更することはできません。
- 表示されているキーワードと関数はAltova XQuery エンジンによりサポートされているものです。
- 関数はXQuery ドキュメント内で定義されたものです。テキストビューにて $ と文字列が入力された場合、その文字列が変数入力ヘルパーに表示されます。入力された変数が既に存在する変数名と同一の場合、新たに入力された変数名が入力ヘルパーから削除されます。
- 入力ヘルパーにてナビゲーションを行うには、入力ヘルパー内のアイテムをクリックして、スクロールバー、マウスホイール、または page-down や page-up キーを使って、リストを移動することができます。

入力ヘルパーにて表示されているアイテムをドキュメント内に挿入するには、挿入を行ないたい場所にカーソルを移動させ、アイテムをダブルクリックします。XQuery では文字列によってキーワードと関数の両方を表す場合があります(例: empty、unordered または except)。これらの文字列は(たとえ関数入力ヘルパーからその名前の関数を選択しても)常に(青色の)キーワードとして入力されます。関数が青色で表示された場合、その名前の後に続くかっこにより、関数と認識させることができます。

8.1.3 XQuery の構文による色分け

XQuery ドキュメントはXQuery コードとXML コードを内部に含むことができます。このセクションではデフォルトのXQuery 構文による色分けについて説明します。XQuery ドキュメント内のXML コードに対して使用される構文色分けは、通常のXML ドキュメントに対するものと変わりません。構文による色分けの色は(XQuery ならびにXML コードに対して)全てオプションダイアログ(「**ツール | オプション**」)のテキストフォントセクションにて設定することができます。(以下のスクリーンショットの例にあるように)中かっこ { } を使うことで、XML 要素内部にXQuery コードを埋め込めることに注意してください。



XQuery モードのXQuery コード内では、以下の構文色分けが使用されます:

- (: コメントは区切り文字も含めて緑色で表示されます :)
- XQuery キーワードは青色で表示されます: キーワード
- XQuery 変数は、ドル記号も含めて紫色で表示されます: $start
- かっこを除く XQuery 関数はオリーブ色で表示されます: function()
- 文字列はオレンジ色で表示されます: "Procedure"
- パス表現など、その他のテキストは(以下の下線部分にあるように)黒色で表示されます:
  for $s in doc("report1.xml")//section[section.title = "Procedure"]
  return ($s//incision)[2]/instrument

これらの色やフォントプロパティはオプションダイアログ(「ツール | オプション」)のテキストフォントセクションで変更することができます。

注意:上のスクリーンショットではコメント部分にあるかっこのペアの1つが太字の黒色で表示されていますが、これはかっこのマッチング機能によるものです(XQuery インテリジェント編集 338 を参照ください)。

8.1.4 XQuery インテリジェント編集

テキストビューのXQuery モードでは以下のインテリジェントな編集機能が提供されます。

- かっこのマッチング 338
- キーワード 338
- 変数 338
- 関数 339
- ビジュアルなガイド 339

かっこのマッチング

かっこのマッチング機能により、かっこのペアとなる開始ならびに終了かっこがハイライトされ、かっこ内に含まれるコンテンツを簡単に確認することができます。この機能は、特にXQuery コメント内にてかっこがネストされているような状況で役に立ちます。

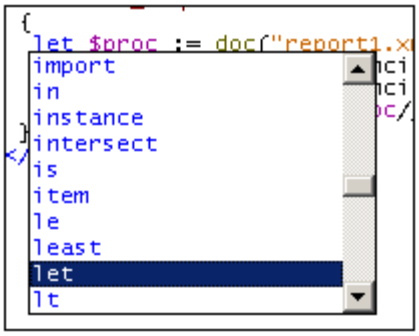
```
1 (:(:(: Filename: seqQ5b.xq :)
2 (: Source: http://www.w3.org/TR/xquery-use-cases :):)
```

- かっこのマッチングはカーソルが(開始または終了)かっこの直前または直後に位置しているときに有効になります。かっこは対となるかっこと一緒に(太字に黒色で)ハイライトされます。上のスクリーンショットにあるカーソルの位置に注目してください。
- かっこのマッチングは、丸かっこ ()、角かっこ [] 、そして中かっこ {} で有効になります。例外は、XML タグに使用される不等号記号 <> です。

メモ: XML 要素の開始または終了かっこのすぐ内にカーソルがあると、両方のかっこがハイライトされます。「Ctrl」 + E のキーを押すことで、対となるペアがハイライトされます。「Ctrl」 + E を連続して押下することで、カーソルの位置を開始ならびに終了かっこ間で切り替えることができます。この機能を使うと、ドキュメントのナビゲーションを素早く行うことができます。

キーワード

XQuery キーワードはクエリー表現で使用される命令で、青色で表示されます。カーソルをキーワードの内部、直前、または直後に移動させることでキーワードを選択できます。キーワードが選択された状態で、「Ctrl」 + 「Space」 を押すことにより、キーワードのリストがポップアップで表示されます。リストをスクロールし、目的のキーワードをダブルクリックすることで、選択されていたキーワードを置き換えることができます。

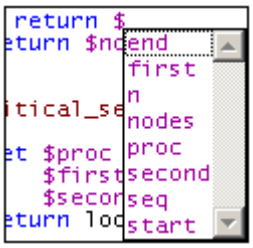


上のスクリーンショットでは、カーソルが let キーワード上に位置されており、ポップアップにあるキーワードをダブルクリックすることで、let キーワードを置き換えることができます。

変数

変数名の前には $ 記号が配置され、紫色で表示されます。このインテリジェント編集機能の仕組みはキーワードの編集機能のそれと似たようなものです。ドキュメント内の変数リストが表示されるポップアップへアクセスするには2つの方法があります:

- $ 文字を入力後に Ctrl+Space を押下
- 変数を選択後に Ctrl+Space を押下（$ 文字の直後、変数名の内部や直後にカーソルを位置させることで、その変数が選択された状態になります）

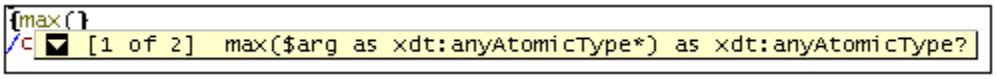


$ 記号の直後に変数を挿入する、または選択された変数を置き換えるには、ポップアップメニュから目的の変数をダブルクリックしてください。

関数

キーワードならびに変数の場合と同様に、（オリーブ色で表示される）関数を選択するか、「Ctrl」+「Space」を押下することで内蔵の関数がポップアップメニューにより表示されます。関数内、または関数名の直前、直後にカーソルを位置させることで、関数が選択された状態になります。関数名の直後にあるかっこ内部にカーソルを位置させても選択されている状態にはなりません。ポップアップメニュー内の関数名をダブルクリックすることで、選択された関数名をポップアップメニューの関数に置き換えることができます。

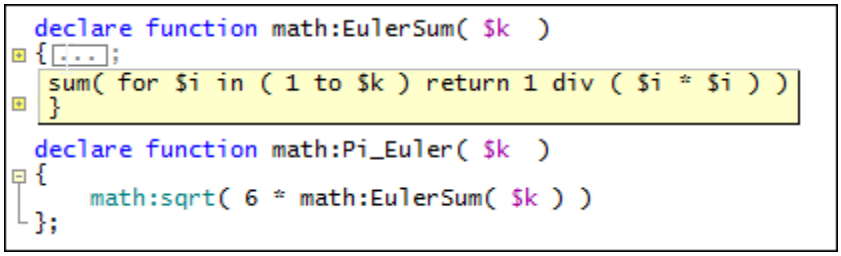
（以下のスクリーンショットにあるように）関数の開始かっこの直後にカーソルをおき、「Ctrl」+「Space」を押下することで、その関数のヒントを表示することができます。標準 XQuery 関数でしかヒントは表示され無いことご注意してください。



ヒントの先頭にある下向きの矢印は、異なる引数や戻り値の型を持つ同名の関数が存在することを表しています。矢印をクリックすることで次の関数を表示することができ、クリックを繰り返すことでその名前の関数を全て表示させることができます。他にも「Ctrl」+「Shift」+「Up」または「Ctrl」+「Shift」+「Down」キーの組み合わせで、同様の操作を行うことができます。

ビジュアルなガイド

テキストの折りたたみ（またはソースのおりたたみ）はXQuery 中かっこ、XQuery コメント、XML 要素、XML コメントにより有効化され、これらのノードを展開または折りたたむ機能を意味します。このようなノードは、+/- サインによりソース折りたたみ余白内で表示されます（*下のスクリーンショット参照*）。余白はテキストビュー設定ダイアログ 969 により切り替えのオンとオフを行うことができます。ノードが折りたたまれると、視覚的に省略記号により表示されます（*下のスクリーンショット参照*）。マウスのカーソルが省略にポイントされると、ポップアップで折りたたまれたノードの内容が表示されます（*スクリーンショット参照*）。内容がポップアップでの表示に大きすぎる場合、ポップアップのしたの部分に省略記号で表示されます。



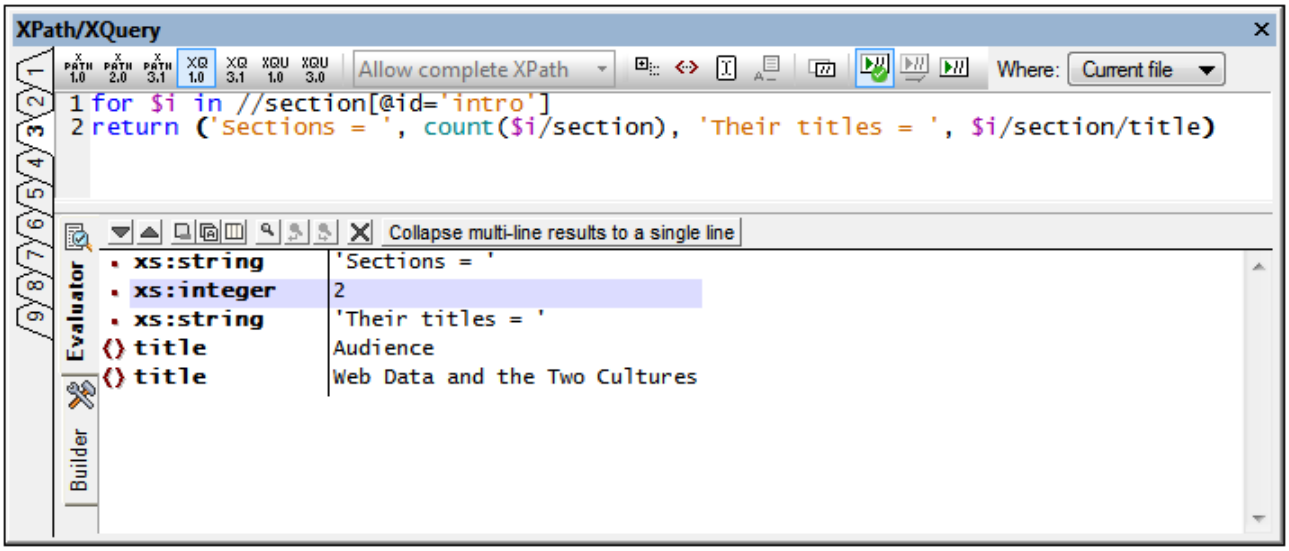
テキストツールバー内の「全ての折りたたみを切り替える」アイコン は、全てのノードを展開された書式でまたはトップレベルのドキュメント要素まで折りたたみ表示します。
ノードの+/- アイコンをクリックすることにより以下のオプションを使用することができます:

Click [-]	ノードを折りたたみます
Click [+]	子孫ノードが展開されて表示されるか、ノードが折りたたまれた前の状態に従い折りたたまれ、ノードが展開されます。
Shift+[-]クリック	すべての子孫ノードを折りたたみ、展開フォーム内でクリックされたノードを残します。
Ctrl+ [+]クリック	クリックされたノードとそのすべての子孫ノードを展開します。

(C) 2015-2021 Altova GmbH

8.2 XQuery 評価

XQuery 式は、XPath/XQuery 出力ウィンドウ内の1つまたは複数のドキュメントに対して評価されることができます（*下のスクリーンショット*）。



以下をおこなってください：

1. ウィンドウの上のペインにXQuery 式 を入力します。
2. *Where* コンボボックス内から（*上のスクリーンショット参照*）クエリされるXMLドキュメントの場所を選択します。 オプション: (i) 現在のファイル; (ii) ファイルを開く; (iii) プロジェクト; (iv) フォルダー.
3. 「XPath/XQuery 式の評価」(F5) をクリックします。XML ファイルに対して式が評価されます。指定された（*Where*）ロケーションが、1 つ以上のXML ファイルを含む場合、XML ファイル内で、式に一致するデータ構造またはコンテンツが検索されます。使用可能なすべての一致は、下のペイン内に表示されています。

上のスクリーンショット内で、属性 `@id='intro'` を持つセクション要素のためにクエリが作成されます。クエリは、この `intro` セクション サブセクションの数量とタイトルを返します。

詳細に関しては、出力ウィンドウ XPath/XQuery 108 および アップデートのプレビューと適用 345 を参照してください。

8.3 XQuery 検証

XQuery または XQuery Update ドキュメントを検証するには以下を行います:

1. XQuery ドキュメントをアクティブなドキュメントにします。
2. 「XML | 検証」を選択、「F8」キーを押す、または「検証」ツールバーアイコンをクリックします。

検証ツールバーアイコン

XQuery 構文に対してドキュメントが検証されます。

(C) 2015-2021 Altova GmbH

8.4 XQuery/Update 実行

XQuery または XQuery Update ドキュメントは以下の方法で実行することができます:

- XQuery または XQuery Update ドキュメントがアクティブな場合。
- XML ドキュメントがアクティブな場合。

メモ　ドキュメントが XQuery ドキュメントまたは XQuery Update ドキュメントであるかは、ドキュメントのファイル拡張子によって決定されます。XMLSpy は、ファイルの種類の関連付けをオプションダイアログのファイルの型セクションによる定義に従い認識します。(ツール | オプション | ファイルの型 1018)。

メモ　XQuery Update には、XPath/XQuery 出力ウィンドウにアップデート式を入力することができます。アップデートが受け入れられた場合、アップデートを適用して、アップデートされたファイルを保存します。詳細に関しては、XQuery Update Facility 345 とアップデートのプレビューと適用 345 を参照してください。

XQuery または XQuery Update ドキュメントを実行する

XQuery または XQuery Update ドキュメントを「*XQuery / XQuery Update ドキュメントをアクティブにする*」で実行する場合は以下を行います。

1. XQuery または XQuery Update ドキュメントをアクティブなドキュメントにします。
2. 「**XSL/XQuery | XQuery/Update 実行**」を選択または、コマンドのツールバーアイコンをクリックします。これにより XQuery のための XML ソースを定義するダイアログが表示されます(下のスクリーンショット)。

XQuery/Update 実行ツールバーアイコン

3. XML ファイルを参照するか、XML ソースの選択をスキップします。

> 通常、XQuery ドキュメントは、単一のドキュメントに関連付けられていません。これは、XQuery 式が `doc()` 関数を使用して、XML ドキュメントをいくつも選択することができるからです。XMLSpy 内では、しかしながら、個別の XQuery ドキュメントを実行する前に、実行のためのソース XML ドキュメントを選択することができます。この場合、選択された XML ソースのドキュメントノードは、XQuery ドキュメントのルートレベルで使用することのできるコンテキストメニューから開始します。行頭スラッシュから開始するパスは、コンテキストアイテムのドキュメントノードとしてこのドキュメントノードと共に解決されます。

メモ　「**XQuery/Update**」実行コマンドは、プロジェクトウィンドウ 105 アイテムのコンテキストメニューから使用することができます。

実行 / アップデートの結果

- *XQuery 実行:* 実行の結果は一時ファイルとして生成され、希望するファイル形式やファイル拡張子で任意の場所に保存することができます。

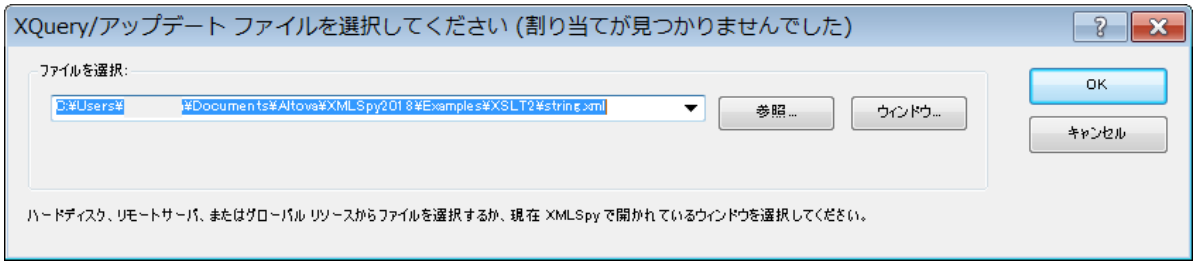
- *XQuery アップデート:* アップデートはファイルに保存されるか、アップデートされたファイルが開かれると、プレビューし保存または閉じることができます。どちらのアクションを実行するか指定することができます。これはオプションダイアログの *XQuery* セクション(「ツール | オプション | XQuery」[1038])で行うことができます。

アクティブな XML ドキュメントでの実行

XQuery または XQuery Update ドキュメントをアクティブなドキュメントで実行するには以下を行います。

1. XML ドキュメントをアクティブにします。
2. 「XSL/XQuery | XQuery/Update 実行」を選択するか、ツールバーアイコンをクリックします。これにより XQuery のための XML ソースを定義するダイアログが表示されます。「XSL/XQuery | XQuery/Update 実行」を選択または、コマンドのツールバーアイコンをクリックします。これにより XQuery のための XML ソースを定義するダイアログが表示されます(下のスクリーンショット)。

   XQuery/Update 実行ツールバーアイコン



3. XQuery または XQuery Update ファイルを参照して「OK」をクリックします。

実行 / アップデートの結果

- *XQuery 実行:* 実行の結果は一時ファイルとして生成され、希望するファイル形式やファイル拡張子で任意の場所に保存することができます。
- *XQuery アップデート:* アップデートはファイルに保存されるか、アップデートされたファイルが開かれると、プレビューし保存または閉じることができます。どちらのアクションを実行するか指定することができます。これはオプションダイアログの *XQuery* タブ(「ツール | オプション | XQuery」[1038])で行うことができます。

XQuery 変数

Altova XQuery エンジンを使用している場合、XQuery 変数を便利な GUI ダイアログ内に補完することができます。XMLSpy を使った XQuery ドキュメントの実行時に、全ての保管された変数は XQuery ドキュメントに渡されます。より詳しい情報については、XSLT パラメーター / XQuery 変数[890] コマンドの記述を参照ください。

Altova XQuery エンジン

Altova XQuery エンジンの実装や XQuery ファイルの処理に関する詳細については、XQuery エンジンの実装[1482] を参照ください。

8.5 XQuery Update Facility

XQuery Update Facility は、ドキュメントの部分を変更することのできる XQuery 言語の拡張機能です。通常の XQuery 実行では、ドキュメント全体が再生成され、そして、元の場所に保管される必要があります。これはドキュメントの小さいパートのみが変更される場合、非効率になる可能性があります。Update Facility を使用することにより変更された部分のみを更新することができます。

XQuery Update Facility は、XQuery 1.0 および XQuery 3.1 の拡張機能として次の仕様内で説明されています:

- XQuery Update Facility 1.0 (2011 年 3 月 17 日版W 3C 勧告)
- XQuery Update Facility 3.0 (2013 年 1 月 8 日版W 3C 作業ドラフト)

XMLSpy 内の XQuery Update Facility

以下の要点は、XQuery Update が XMLSpy 内でどのように作動するかを説明しています:

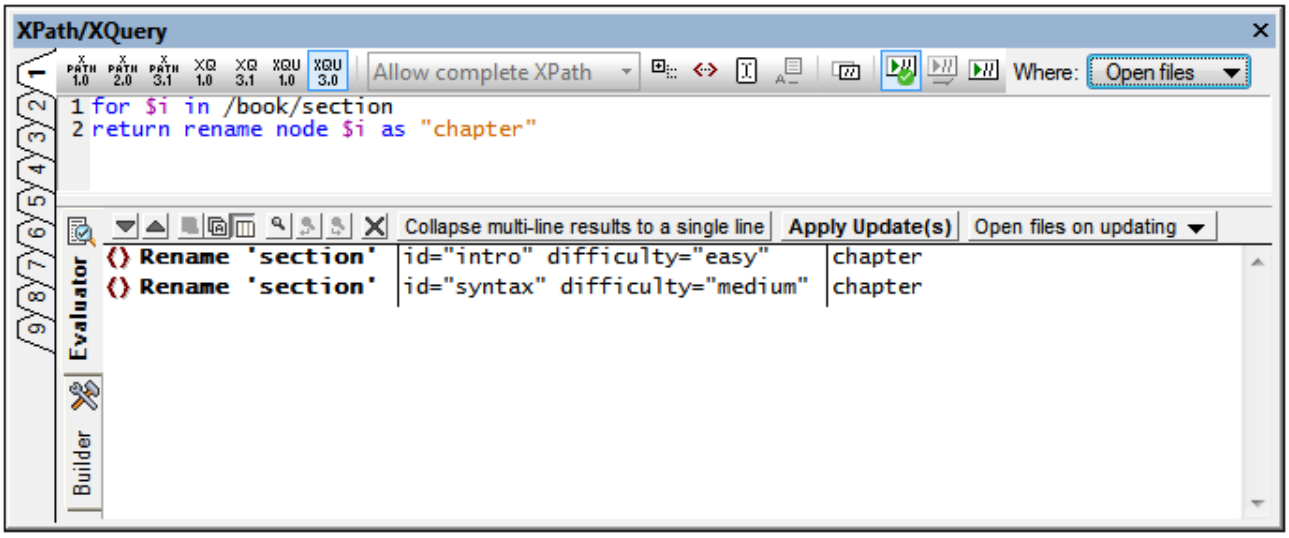
- アップデートはアップデート式により行われます。例えば、アップデート式は、XML ドキュメント内のノードの名前の変更を指定することができます:
  ```
  rename node /documents/doc-01 as "document-01"
  ```
- 実際のところは、複数のアップデート式が単一の XQuery Update ドキュメントに入力されます。
- アップデートドキュメント内の各 アップデート式が実行されると、結果はすぐに適用されませんが、*保留中の アップデート リスト (PUL)*に追加されます。 この結果、PUL は、すべてのアップデート式の結果を含みます。すべての updates in PUL のすべてのアップデートは、すべてに一度に適用されます。
- XMLSpy 内で、PUL アップデート 以下の 2 つの方法で適用されます:
  (i) GUI 内でユーザーによりプレビューする利点は、プレビューは、予期しない結を果表示します。XQuery/Update 実行コマンド[343] *または XPath/XQuery 出力ウィンドウ*[345]*. 内の XQuery アップデート式評価中でプレビューを使用することができます。プレビューオプションに関しては、関連する箇所で説明されています。*
  (ii) 直接 そして ユーザーによる中断なし。利点は、アップデートが警告なしにユーザーによる中断なしに実行されます。アップデートの直接のアプリケーションは XQuery/Update 実行コマンド[343] *または XPath/XQuery 出力ウィンドウ*[345] *内で XQuery アップデート式iの評価を実行中に使用することができます。(プレビューなしの) 直接的アップデートのアプリケーションがが直接のアップデートオプションで使用可能な場合、オプションに関する情報は関連する箇所で説明されています。*

XMLSpy は、アクティブな XML ドキュメントに対して使用されたアップデート式の効果をプレビューすることのできる、パワフルな XQuery Update プレビュー機能を与えます。この機能については、アップデートのプレビューと適用[345] のセクションで説明されています。

8.5.1 アップデートのプレビューと適用

XQuery Update を使用して、XML ドキュメントを修正する場合、XML ドキュメントに適用する前にプレビューして、変更されたドキュメントを保存することができます。

XPath/XQuery 出力ウィンドウ(*下のスクリーンショット*)内に 1 つまたは複数の アップデート式を入力することができます、そして下のペインに表示された **保留中のアップデートリスト (PUL)** 内でアップデートをプレビューすることができます(*下のスクリーンショット参照*)。PUL が希望する状態の場合、ドキュメントのアップデートを適用し、変更されたドキュメントを保存することができます。PUL に変更を付け加える場合は、変更を適用しない、または保存しないことを選択できます。

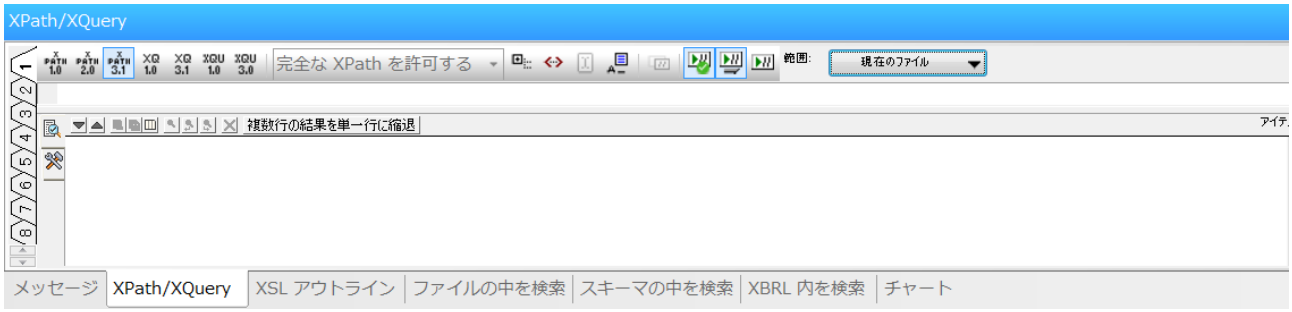


アクティブな XML ファイルのためにPUL を作成するには、次を行います:

1. XPath/XQuery 出力ウィンドウのツールバーから(*上のスクリーンショット*) XQU 1.0 または XQU 3.0 アイコン(それぞれ XQuery Update 1.0 または XQuery Update 3. 0 のため) に選択します。
2. 1 つまたは複数のアップデート式をウィンドウの上のペインに入力します。アップデート式と構文の説明に関しては、操作と構文のアップデート 347 のセクションを参照してください。
3. ツールバーのスキャンロケーションコンボボックスから、アップデートのためにスキャンされるロケーションを選択します:
   ファイルを開く: XMLSpy で現在開かれているすべてのファイルがスキャンされます。
   現在のファイル: 現在アクティブな ファイルのみがスキャンされます。スキャンのために選択されたロケーションが*現在のファイル*の場合、「XPath/入力中に XQuery 式の評価する」ツールバーアイコンが有効化されます。
   プロジェクト: 現在アクティブなプロジェクトがスキャンされます。
   フォルダー: スキャンするフォルダーを選択することができます。
4. アップデート 式 を実行して、PUL を表示するには、「XPath/XQuery 式の評価」ツールバーアイコンをクリックします。

XPath/XQuery 出力ウィンドウツールバー

以下のXPath/XQuery 出力ウィンドウツールバーコマンド(*下のスクリーンショット内で赤枠の中*) を使用することができます。

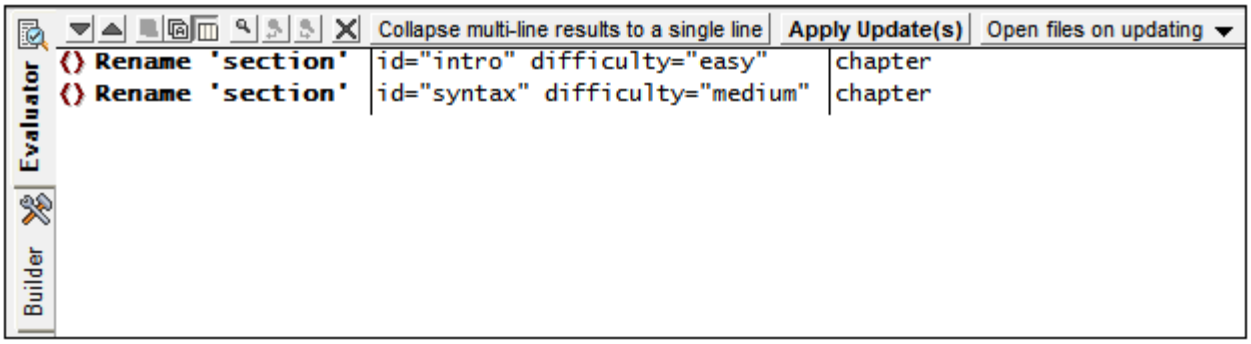


- ヘッダーの表示は、アップデートロケーションのファイル名の表示の切り替えを行います。
- 完全な結果の表示は、ノードコンテンツ全体またはノードの属性のみの表示の切り替えを行います。
- XPath/XQuery 起点の設定は、アクティブなファイル内のカーソルロケーションを相対的な XPath 式の起点として設定します。式 は、このノードに対して相対的に評価されます。

- 手動でノードのロケーターパス式を入力するかわりに、XMLSpy が入力を行うことができます。これは以下のように行います: (i) XPath 式にロケーターパスを入力する箇所にカーソルをポイントします、(ii) ターゲットするノードの開始タグ内にカーソルをポイントします、(iii) 式内にロケーターパスを入力するために、編集フィールドアイコン の現在の選択の XPath のコピーをクリックします。
- XML の検証 ファイルは、XML ファイル検証の切り替えを行います。
- 入力中の XPath/XQuery 式の評価コマンドは、解析オプションが現在のファイルに設定されていると有効化されます。コマンドがオンに切り替えられている場合、アップデート式が入力されている場合でも PUL は、生成されます。
- XPath/XQuery 式の評価 コマンドは、アップデート式を評価し、PUL を生成します。
- ロケーションのスキャン コンボボックス オプションは、上で説明されています。

保留中のアップデートリスト (PUL) ペイン

PUL ペインは、実行されるすべてのアップデートを表示します。ヘッダーの表示オプションがウィンドウのツールバー内で切り替えられている場合、ターゲットファイルのロケーションが表示されます。PUL の表示は、3 つの垂直のセクションに分割されています (*下のスクリーンショット参照*): (i) 実行するアップデート アクション、(ii) ターゲットノードのコンテンツはアップデートされます。(iii) アップデートアクションの結果



次の PUL ペイン ツールバーコマンドを使用することができます:

- **次と前** アイコンは、現在選択されているメッセージの次および前のメッセージをそれぞれ選択します。
- **ラインのコピー コマンド** は、選択されたラインおよびすべてのラインをクリップボードにそれぞれコピーします。
- **検索** コマンドは、PUL ペイン内のテキストを検索します。
- **クリア** コマンドは、PUL ペイン内のすべてのラインを削除します。
- **複数ラインの結果のおりたたみ** コマンドは、複数のラインのアップデートアイテムを単一のラインに折りたたみます。
- **アップデートの適用** コマンドは、保留中のアップデートをターゲットの場所に適用します。アップデートする際に、アップデートは、ファイルに保存、または、アップデートされたファイルが表示されます (そして、手動で 保存されるかを決定することができます)。次のオプションを参照してください。
- アップデートファイル ディレクトリコンボボックスにより以下を選択することが許可されます (i) ファイルがダイアログを表示せずにディスクにアップデート、または (ii) アップデートされたファイルが開かれそしてアクティブになるか。後者のオプション が選択された場合、非-アクティブなターゲット ファイルはアクティブに設定されます。変更されたドキュメントを保存または保存しない オプションがあります。

メモ 1 つまたは複数のファイルが直接ディスクにアップデートされた場合、変更されたファイルのリストが表示されます。リストの各ラインは、ファイルのロケーションおよびファイルへのクリックすることのできるリンクを表示します。

8.5.2 Update オペレーションと構文

XQuery Update Facility では以下のオペレーションを使用することができます:

- 1 つまたは複数のノードの削除 348 する

- 1つまたは複数のノードを指定されたノードの前、後、または内部に挿入 348 する
- ノードの名前の変更 349 をする
- アイテムのシーケンスのノードと置換 349 える
- アイテムのシーケンスの文字列の値とノードの値を置換 350 える

これらのオペレーションのキーワード及び構文はこのセクションのサブセクションで説明されています。

8.5.2.1 ノードの削除

説明と構文
1つ以上のノードを削除します。

```
delete node nodeSequence
delete node nodeSequence
```

詳細

- `nodeSequence` 式は削除するノードのシーケンスを返します。選択されたノードは削除されるためマークされます。
- 単一 ノード または複数の ノード が使用されることができます。`nodeSequence` 内のアイテムの数に対するコレスポンデンスは必要ありません。

サンプル

```
for $i in /book/section return
delete nodes $i/@id
```

8.5.2.2 ノードの挿入

説明と構文
1つまたは複数のノードを指定されたターゲットノードの前、後、または内部に挿入する。

```
insert (node|nodes) items into targetNode
insert (node|nodes) items as first into targetNode
insert (node|nodes) items as last into targetNode
insert (node|nodes) items before targetNode
insert (node|nodes) items after targetNode
```

詳細

- `items` 式は、アイテムのシーケンスを返さなければなりません。キーボード `node|nodes` が使用されていますが、`items` は、非ノードアイテムのシーケンスである場合があります。
- `targetNode` 式は、単一のターゲットノードを指す必要があります。
- `into` キーボードが使用されていますが、`targetNode` は要素ノードまたはドキュメント要素ノードである必要があります。
- キーフレーズ `as first` または `as last` が使用されていますが、挿入は、最初または最後の子です。
- キーワード `into` は単独で仕様され、属性は既存の属性に追加され、要素は最初の子として挿入されます。
- キーワード `before` または `after` が使用されており、`targetNode` はあらゆる型であることができます。
- 属性が挿入される場合、既存の属性の名前を重複してはなりません。

サンプル

```
for $i in /book/section return
insert nodes (attribute id { 'somevalue' }, <newelement>some content including the
numbers "{ 1 to 3}"</newelement>)
into $i
```

8.5.2.3 ノードの名前の変更

説明と構文
要素、属性、または指示ノードの名前を変更する。

```
rename node targetNode as name
```

詳細

- 要素、属性、または処理命令であることのできる *targetNode* 式は単一ターゲットノードを指す必要があります。
- *name* 式は、QName または文字列を評価する必要があります。
- QName が作成されると、必須名前空間がローカルで宣言されます。

詳細

```
rename node /book/title as 'header-1'

rename node /book/title as QName("http://www.altova.com/xquf", "header-1")
```

8.5.2.4 ノードの置換

説明と構文
アイテムのシーケンスとノードを置き換えます。

```
replace node targetNode with items
```

詳細

- *targetNode* 式は、単一のターゲットノードを指す必要があります。
- *items* 式はアイテムのシーケンスを返す必要があります。このシーケンスはターゲットノードを置き換えます。
- 属性ノード以外、ターゲットノードはどの型のシーケンスとも置換えられることができます。
- 属性ノードは属性ノードと置き換えられます。下のサンプルを参照してください。

サンプル

```
replace node //hr with '<line/>'

for $i in //@height return
replace node $i with (attribute line-height{'12pt'})
```

8.5.2.5 ノードの値の置換

説明と構文
アイテムのシーケンスの文字列の値とノードの値を置き換えます。

```
replace value of node targetNode with items
```

詳細

- `targetNode` 式は、単一のターゲットノードを指す必要があります。
- `items` 式はアイテムのシーケンスを返す必要があります。
- ターゲットノードのコンテンツは `items` 式により返されたシーケンスの文字列の値と置き換えられます。これは、ターゲットノードはテキストノード1つのみを含むことを意味します。

サンプル

```
for $i in //title return
replace value of node $i with ('Draft Title')
```

8.5.2.6 fn:put 関数

fn.put 関数はXQuery 内蔵関数ライブラリの拡張子としてQuery Update Facility 1.0 により与えられています。（ **fn:** このセクションの名前空間のプレフィックスは、以下の名前空間にバインドされているものと見なされます: `http://www.w3.org/2005/xpath-functions`.)

```
fn:put($node as node(), $uri  as xs:string) as empty-sequence()
```

関数は、ドキュメントまたは要素を `$uri` により指定された場所に保管します。ファイルシステムまたはデータベースなどの外部ストレージシステムのリソースを作成するために呼び出されます。`fn:put` の外部効果は、XQuery のドメイン外で起こるため、実装定義済みです。目的は、`fn:put` がドキュメントノードにより呼び出され、エラーが指摘されなかった場合、次のクエリは、同じ URI を持つ `fn:doc` を呼び出して保管されたドキュメントにアクセスすることができます。

詳細は仕様を参照してください。

8.6 XQuery と XML データベース

XQuery ドキュメントを使えば、XML データベース（XML DB）への問い合わせを行うことができます。このXQuery 機能は、現在 IBM DB2 データベースでしかサポートされておりません。XQuery を使ったXML DB への問い合わせの仕組みは、(i) XQuery エンジンに対してDB 内部のXML へ問い合わせを行う通知と、(ii) DB 内部にあるXML データへのアクセス、により成り立っています。

この仕組みを実装するには、以下のようなステップを踏む必要があります。各ステップの詳細を以下に記します:

1. XQuery ドキュメントの設定 351 を行い、XQUERY キーワードをドキュメントの最初に挿入することでXML DB の問い合わせを行うようにする。.
2. アクティブな XQuery ドキュメントに対して、（情報ウィンドウから）DB のサポートを有効 351 にして、（クイック接続ダイアログを使って）DBへ接続する 351 。
3. XQuery ドキュメントにて、DB 固有のXQuery 拡張 351 を使用することで、DB データにアクセスし、XQuery オペレーションも行えるようにする。
4. XQuery ドキュメントの実行 351 をXMLSpy から行う。

XML DB の問い合わせを行うよう XQuery ドキュメントを設定する

XQuery ドキュメントを使ってXML DB の問い合わせを行うには、XQuery ドキュメントを開き（または新しいXQuery ドキュメントを作成し）、XQUERY キーワードを、ドキュメントの最初に入力します。以下の例を参照ください。

```
XQUERY (: Retrieve details of all customers :)
declare default element namespace "http://www.altova.com/xquery/databases/db2";
<a> {db2-fn:xmlcolumn("CUSTOMER.INFO")} </a>
```

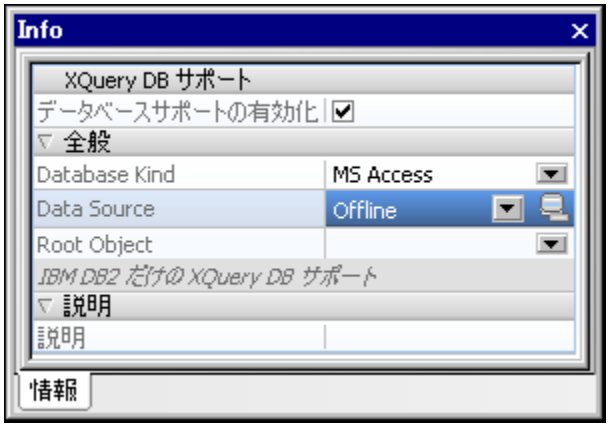
ドキュメントがオプションのxquery version 表現を使用する場合でも、XQUERY が必要です:

```
XQUERY xquery version "1.0"; (: Retrieve details of all customers :)
declare default element namespace "http://http://www.altova.com/xquery/databases/db2";
<a> {db2-fn:xmlcolumn("CUSTOMER.INFO")} </a>
```

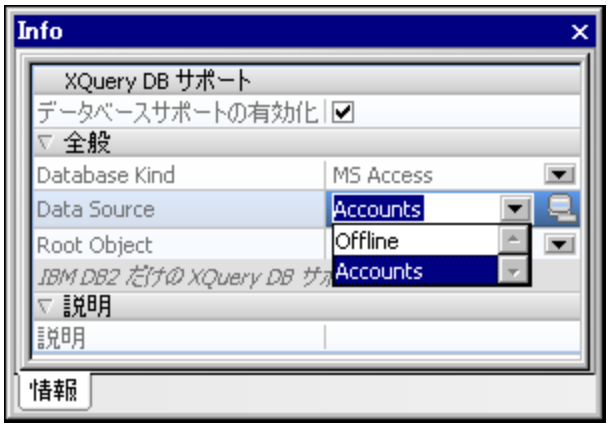
メモ　XMLSpy の内蔵 XQuery エンジンがXQUERY キーワードを読むことにより、XML DB がアクセスの対象になることが示唆されます。その結果、XQUERY キーワードを含むXQuery ドキュメントを、XML DB を含まないXML ドキュメントに対して実行してもエラーとなります。

DB サポートを有効化し、DB へ接続する。

（下のスクリーンショットの様に）情報ウィンドウにあるデータベースサポートを有効化チェックボックスにチェックを入れることで、XQuery ドキュメントのDB サポートが有効になります。DB サポートは、各 XQuery ドキュメントに対して有効にする必要があり、XQuery ドキュメントを開き直したときにも再度設定する必要があることに注意してください。



情報ウィンドウにてDB のサポートを有効にすると、接続ウィザードが起動され、データベースへの接続を行うことができます。現在のところ、IBM DB2 データベースのみがサポートされています。データベースへの接続方法についてはデータソースへ接続 573 、のセクションにて記述されています。データソースへの接続が既に存在する場合、これらの接続が(以下のスクリーンショットにある)情報ウィンドウのData Source コンボボックスに表示され、データソースのリストに表示されているアイテムの1つを、アクティブな XQuery ドキュメントのデータソースとして選択することができます。情報ウィンドウでは、Root Object コンボボックスからルートオブジェクトを選択することもできます。



情報ウィンドウにある アイコンをクリックすることで、(DB への接続を行う)接続ウィザードにいつでもアクセスすることができます。

メモ XQuery ドキュメントを閉じると、DB への接続も同様に切断されます。その後 XQuery ドキュメントを開く際に、DB への接続を再度行う必要が生じます。

IBM DB2 固有のXQuery 言語拡張

XQuery ドキュメントではIBM DB2 データベースからデータを取得するために、2つのIBM DB2 固有の関数を使用することができます:

- `db2-fn:xmlcolumn` により、列の検索やフィルタリングを行うこと無く、XML 列全体を取得することができます。
- `db2-fn:sqlquery` により、SQL `SELECT` ステートメントをベースにした値の取得を行ないます。

これらの関数を使用して取得された XML データは、標準的な XQuery コンストラクトにより開くことができます。以下の例を参照ください。

`db2-fn:xmlcolumn`: この関数の引数は大文字/小文字で区別される文字列リテラルで、テーブル内にある XML 列の参照に使用されます。文字列リテラル引数は XML タイプの修飾された列名でなければなりません。列内にある全ての XML データがシーケンスとして関

数から返され、ここでは検索条件などは適用されません。以下の例では、CUSTOMER テーブル内の INFO (XML) 列のデータ全てが、トップレベルの<newdocelement> 要素内に戻されます:

```
XQUERY (: Retrieve details of all customers :)
declare default element namespace "http://www.altova.com/xquery/databases/db2";
<newdocelement> {db2-fn:xmlcolumn("CUSTOMER.INFO")} </newdocelement>
```

取得されたデータは XQuery コンストラクトを使って絞り込むことができます。以下の例では、CUSTOMER テーブルにある INFO (XML) 列から取得されたデータが、XQuery コンストラクトによりフィルタリングされ、トロント (Toronto) にある顧客データのみが取得されるようになっています。

```
XQUERY (: Retrieve details of Toronto customers :)
declare default element namespace "http://www.altova.com/xquery/databases/db2";
<newdocelement> {db2-fn:xmlcolumn("CUSTOMER.INFO")/customerinfo[addr/city='Toronto']}
</newdocelement>
```

メモ　上の例では、各セルにある XML ファイルのドキュメント要素は customerinfo であり、db2-fn:xmlcolumn により得られた XML シーケンスのルートノードは customerinfo ノードの上にある abstract ノードと見なされます。

db2-fn:sqlquery:.この関数は SQL SELECT ステートメントを引数として受け取り、その結果を XML 値のシーケンスとして戻します。取得されたシーケンスは XQuery コンストラクトにより絞り込むこともできます。以下の例では、CUSTOMER テーブルにある INFO 列のデータのうち、CID フィールドの値が 1000 から 1004 間にある行が取得されます。SQL ステートメントでは大文字/小文字の区別は行われませんが、XQuery では異なるものとして扱われることに注意してください。

```
XQUERY (: Retrieve details of customers by Cid:)
declare default element namespace "http://www.altova.com/xquery/databases/db2";

<persons>
   {db2-fn:sqlquery("SELECT info FROM customer WHERE CID>1000 AND CID<1004")/
   <person>
      <id>{data(@Cid)}</id>
      <name>{data(name)}</name>
   </person>}
</persons>
```

上の XQuery ドキュメントは以下のような出力を生成します:

```
<persons xmlns="http://www.altova.com/xquery/databases/db2">
   <person>
       <id>1001</id>
       <name>Kathy Smith</name>
   </person>
   <person>
       <id>1002</id>
       <name>Jim Jones</name>
   </person>
   <person>
       <id>1003</id>
       <name>Robert Shoemaker</name>
   </person>
</persons>
```

以下の点に注意してください:

- デフォルトの要素名前空間宣言(デフォルトの element namespace)は XQuery 全体に対して適用され、XML ドキュメントや新たな要素の構築時のナビゲーションに利用されます。これは、XQuery セレクター name が、<default-element-namespace>:name に拡張され、構築された要素(例:persons)がデフォルトの名前空間にあることを意味します。

- SQL SELECT ステートメントでは大文字/小文字の区別が行われません。
- SELECT ステートメントの `WHERE` 句は、データベースアイテムを参照するもので、アクセスされている XML ファイルのノードを参照するものではありません。
- `db2-fn:sqlquery` 関数の後にある "/" は戻されたシーケンスの最初のアイテムを表わしており、このアイテムがコンテキストノードになります。

XQuery の実行

XQuery ドキュメントを実行するには、XSL/XQuery メニューにある XQuery を実行コマンドを選択します。他にも、Alt+F10 を押下し、XQuery を実行アイコン をクリックすることで、同様の操作を行えます。実行の結果は新しいドキュメント内に表示されます。

9 XSLT/XQuery デバッガー とプロファイラー

XMLSpy には、ドキュメントを速く作成する手助けをする XSLT/XQuery デバッガー 356 が搭載されています。

これらの2 つの機能に関しては、このセクションのサブセクションで説明されています。

9.1 XSLT と XQuery デバッガー

Altova Web サイト: XSLT デバッガー、XQuery デバッガー

XSLT and XQuery Debuggerを使ってXSLT スタイルシートやXQuery ドキュメントの試験やデバッグを行うことができます。XSLT and XQuery Debuggerのインターフェースにはお互いに連動するXSLT/XQuery ドキュメント、結果ドキュメント、そしてソースXML ドキュメントのビューが搭載されており、XSLT/XQuery ドキュメントをステップ毎に実行することができます。対応する出力もステップごとに生成され、ソースXML ファイルが表示されていれば、XML ファイル内の対応する箇所が各ステップでハイライトされます。インターフェース内にあるウィンドウにはデバッグ情報が表示されます。

XSLT and XQuery Debuggerは常にデバッガーセッション内で起動されます。デバッガーセッションは以下のうちのどれかです:

- 内蔵のAltova XSLT 1.0 エンジンを使用するXSLT 1.0
- 内蔵のAltova XSLT 2.0 エンジンを使用するXSLT 2.0
- 内蔵のAltova XSLT 3.0 エンジンを使用するXSLT 3.0
- 内蔵のAltova XQuery 1.0 エンジンを使用するXQuery 1.0
- 内蔵のAltova XQuery 3.1 エンジンを使用するXQuery 3.1

メモ XQuery Update Facility 1.0 とXQuery Update Facility 3.0 は現在 XSLT and XQuery Debugger でサポートされていません。

開かれるデバッガーセッションの種類は、デバッガーセッションが開かれたドキュメント（*アクティブドキュメント* または*アクティブファイル*）の種類により自動的に決定されます。XSLT デバッガーセッションはXSLT ファイルから開かれ（どのバージョンかは、XSLT スタイルシート（XSLT 1.0 には`"1.0"`、XSLT 2.0 には`"2.0"`、XSLT 3.0 には`"3.0"`））の`xsl:stylesheet` や`xsl:transform` 要素にある`version` 属性の値により変化します）、XQuery デバッガーセッションはXQuery ファイルから開かれます。アクティブファイルがXML ファイルであれば、XML ファイル上で動作させるのがXSLT かXQuery かにより、デバッガーセッションの種類が決定されます。前者の場合選択はスタイルシートがXSLT 1.0、XSLT 2.0、またはXSLT 3.0 スタイルシートであるかにより決定されます。

この情報は下のテーブルにまとめられています。

アクティブなファイル	関連したファイル	デバッグ セッション
XSLT 1.0	XML; (必須)	XSLT 1.0 (内蔵のAltova XSLT 1.0 エンジン使用)
XSLT 2.0	XML; (必須)	XSLT 2.0 (内蔵のAltova XSLT 2.0 エンジン使用)
XSLT 3.0	XML またはJSON; (必須)	XSLT 3.0 (内蔵のAltova XSLT 3.0 エンジン使用)
XQuery 1.0	XML; (オプション)	XQuery 1.0 (内蔵のAltova XQuery 1.0 エンジン使用)
XQuery 3.1	XML またはJSON; (オプション)	XQuery 3.1 (内蔵のAltova XQuery 3.1 エンジン使用)
XML	XSLT 1.0, 2.0, または3.0, またはXQuery 1.0 または3.1; (必須)	XSLT 1.0、XSLT 2.0、3.0 またはXQuery 1.0 または3.1 。XSLT スタイルシートの`xsl:stylesheet` (または`xsl:transform`) 要素にある`version` 属性の値により、XSLT スタイルシートを決定。

3つのAltova エンジンに関する詳細は、付録[1481]内のXSLT とXQuery エンジン情報[1482]を参照してください。

Altova RaptorXML 2021 を使用したXSLT とXQuery タスクの自動化

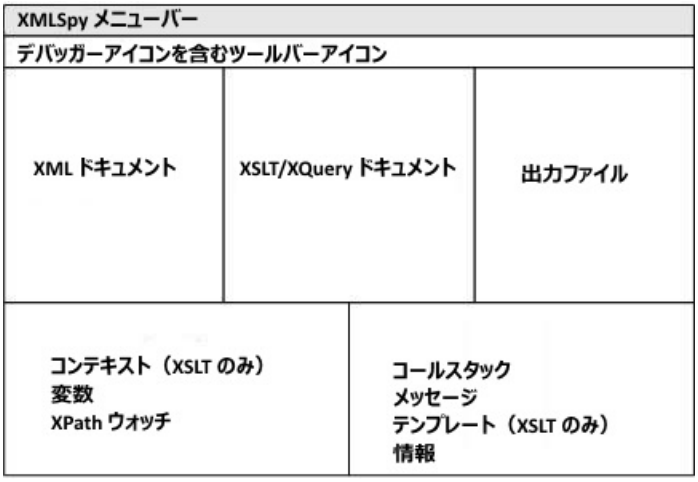
Altova RaptorXML Server は、Altova のXML バリデーター、XSLT 1.0、XSLT 2.0、XSLT 3.0、そしてXQuery 1.0 と 3.1 エンジンを搭載した無料アプリケーションです。コマンドライン、COM インターフェース、Java プログラム、そして.NET アプリケーションからXML ドキュメントの検証を行ったり、XSLT スタイルシートを使ったXML ドキュメントの変換や、XQuery ドキュメントの実行を行うことができます。

RaptorXMLを使えばXSLT ならびにXQuery 処理を自動化することができます。例えば、RaptorXMLを呼び出して、XML ドキュメントのセットを変換し、XQuery ドキュメントのセットを実行するバッチファイルを作成することができます。詳細はRaptorXML Server ドキュメントを参照ください。

9.1.1 メカニズムとインターフェース

XSLT ならびにXQuery ファイルのデバッグにて使用されるXSLT and XQuery Debugger の概要を以下に示します。（「XSL/XQuery | デバッガーの開始/実行」コマンドにより両方とも開始されるものですが、デバッグセッションとデバッガーは異なるものであることご注意してください。まずはデバッグセッションを開始して、デバッガーによりXSLT またはXQuery ドキュメントをステップ実行していく必要があります。

- 「XSL/XQuery | デバッガーの開始/実行」コマンドによりデバッグセッションを開きます。アクティブファイルを元に適切なセッション（XSLT 1.0、XSLT 2.0、XSLT 3.0、XQuery 1.0 またはXQuery 3.1）XSLT とXQuery デバッガー356を参照ください）。XSLT and XQuery Debuggerはテキストビューならびにグリッドビューでのみ動作します。アクティブドキュメントのビューがテキストビューでもグリッドビューでもない状態でデバッグセッションを開始した場合、XSLT and XQuery Debuggerのデフォルトビューであるテキストビューにビューを変更する許可を求められます。デバッガー設定ダイアログ362にて、常に許可を行うよう設定することもできます。
- 「XSL/XQuery」メニューにある「ステップイン」、「ステップアウト」、そして「ステップオーバー」コマンドを使用することでXSLT またはXQuery ドキュメントをステップごとに実行することができます。ここでデバッガーの開始/実行コマンドをクリックすると、ブレークポイントに到達するまで変換または実行を行ないます。ブレークポイントがいつもセットされていない場合、デバッグの結果を表示すること無く、1ステップで全ての変換や実行を行なうことになります。このセッションにXML ファイルが割り当てられている場合、XML ファイルの対応する箇所がハイライトされます。更に、対応するステップにより生成された結果は、結果ファイルに蓄えられ、ステップごとに結果ドキュメントが更新されます。この方法により、XSLT やXQuery ファイルがステートメントごとに行う動作を確認、分析することができます。



上で示されている3つのドキュメント（XML、XSLT/XQuery、出力）の代わりに、2つのドキュメント（XSLT/XQuery、出力）、またはどれか1つのドキュメントビューを選択することもできます。

- デバッグセッションが開いている間、インターフェースの情報ウィンドウには変換/実行に関する様々な情報が表示されます(変数、XPath Watch、コールスタック、メッセージなど)。
- デバッグセッションが開いている間、デバッガーを停止(デバッグセッションは停止していないことに注意)して、ドキュメントを修正することもできます。XMLSpy で使用することのできる編集機能は、デバッグセッション中にファイルを編集する際にも全て使用することができます。デバッガーが停止している間でも XSLT and XQuery Debugger のインターフェースは開かれたままになり、情報ウィンドウに表示されている情報の参照を行うことができます。デバッグセッションにてデバッガーを停止させた後、同じデバッグセッションにて(XSLT/XQuery ドキュメントの初めから)デバッガーを再度開始することもできます。
- XSLT ファイルにブレークポイントをセットすることで、選択された箇所で割り込みを発生させることができます。この機能により、XSLT または XQuery ドキュメント内のステップを手動で個別に実行する必要が無くなり、デバッグセッションの効率を大幅に向上させることができます。
- XSLT にはトレースポイントをセットすることもできます。トレースポイントがセットされた命令にて処理が実行された場合、その命令の値が出力されます。
- デバッグセッションを停止します。この操作により XSLT and XQuery Debugger インターフェースが終了し、元の XMLSpy 環境へ切り替わります。情報ウィンドウにある情報は利用できなくなります。セットされたブレークポイントとトレースポイントに関する情報は、そのファイルが閉じられるまで保持されたままになります(従って、新たなデバッグセッションが開始された場合、新たに表示されるデバッグセッションにもブレークポイントやトレースポイントは表示されます)。

メモ: デバッガーアイコンのあるデバッガーツールバーは、デバッグセッションの開始時に自動的に表示されます。

9.1.2 コマンドとツールバーアイコン

XSL/XQuery メニューやツールバーアイコンからデバッグコマンドへアクセスすることができます。デバッガーアイコンは、デバッグセッションが開始されたときに、ツールバーに自動的に表示されます。以下にデバッガーアイコンを示します。

デバッガーの開始/実行 (Alt+F11)

XSLT/XQuery ドキュメントの処理を開始、または最後まで継続します。ブレークポイントがセットされている場合、そこで処理が停止します。デバッグセッションが開始されていない場合、このボタンによりデバッグセッションが開始され、最初に処理されるノードで停止します。セッションが動作中の場合、XSLT/XQuery ドキュメントはドキュメントの最後または次のブレークポイントまで処理されます。トレースポイントがセットされている場合、(処理命令がそこにたどり着いたときに)各トレースポイントにある命令の値がトレースウィンドウに表示されます。

アクティブなドキュメントのみを表示

デバッガーインターフェースにて現在アクティブなドキュメントのウィンドウを最大化します。

XSLT/XQuery と出力結果を表示

XSLT または XQuery ドキュメントウィンドウを表示し、XML ドキュメントを隠します。

XML、XSLT/XQuery、出力結果を表示

XML、XSLT/XQuery、そして出力ドキュメントを表示します。これはXMLドキュメントがデバッグセッションに割り当てられた時の標準のビューです。

デバッガーを停止

デバッガーを停止します。デバッグセッションの停止とは異なることに注意してください。デバッグを行っている最中にドキュメントの編集を行ったり、同じデバッグセッションにて別のファイルを使用するときに便利な機能になります。デバッガーを停止させた後は、XSLT/XQuery ドキュメントの最初からデバッガーを再度開始させることができます。

ステップイン(F11)

スタイルシートにある全てのノードやXPath 条件式の内部を処理します。このコマンドはデバッガーを再度開始するためにも使用することができます。

ステップオーバー(Ctrl+F11)

現在のノードから同じレベルの次のノードへ移るか、1つ上のレベルにある次のノードに移ります。このコマンドはデバッガーを再度開始するためにも使用することができます。

ステップアウト(Shift+F11)

現在のノードから、次の兄弟ノードに移動するか、1つ上のレベルにある次のノードに移ります。

現在の実行ノードを表示

XSLT/XQuery ドキュメントにある現在の実行ノードを表示、選択し、XML ドキュメント内の対応するノードをハイライトします。他のタブを選択し、XSLT スタイルシートまたはXML ファイルのコードを選択した後、以前作業していた場所を特定するのに便利な機能です。

デバッガーを再起動

出力ウィンドウに表示されている内容を破棄し、現在選択されているファイルでデバッグセッションを再度開始します。

ブレークポイントを挿入/削除(F9)

現在のカーソル位置でブレークポイントを挿入、または削除します。インラインブレークポイントはXSLT/XQuery とXML の両方のノードに対して定義することができ、処理がどこで停止するかを決定することができます。赤色の点線がブレークポイントをセットしたノードの上に表示されます。終了ノードにブレークポイントをセットすることはできず、XSLT ドキュメントの属性に対するブレークポイントは無視されます。このコマンドは右クリックして表示されるコンテキストメニューからも利用することができます。

トレースポイントを挿入/削除(Shift+F9)

現在のカーソル位置でトレースポイントを挿入、または削除します。インライントレースポイントはXSLT ドキュメントのノードに対して定義することができます。デバッグ中にトレースポイントがセットされたステートメントにたどり着くと、そのステートメントの結果がトレースウィンドウに出力されます。青色の点線がトレースポイントをセットしたノードの上に表示されます。終了ノードにトレースポイントをセットすることはできません。このコマンドは右クリックして表示されるコンテキストメニューからも利用することができます。

ブレークポイントの有効/無効化(Ctrl+F9)

このコマンド(ツールバーアイコンにはありません)により、既に定義されたブレークポイントを有効/無効化することができます。ブレークポイントが無効化されると、赤色のブレークポイントが灰色で表示されます。デバッガーは無効化されたブレークポイントでは停止しません。ノード名までカーソルを移動させ、ブレークポイントの有効/無効化コマンドをクリックして、ブレークポイントを有効/無効化することができます。このコマンドはブレークポイントのある位置で右クリックすることでも使用可能です。

トレースポイントの有効/無効化(Shift+Ctrl+F9)

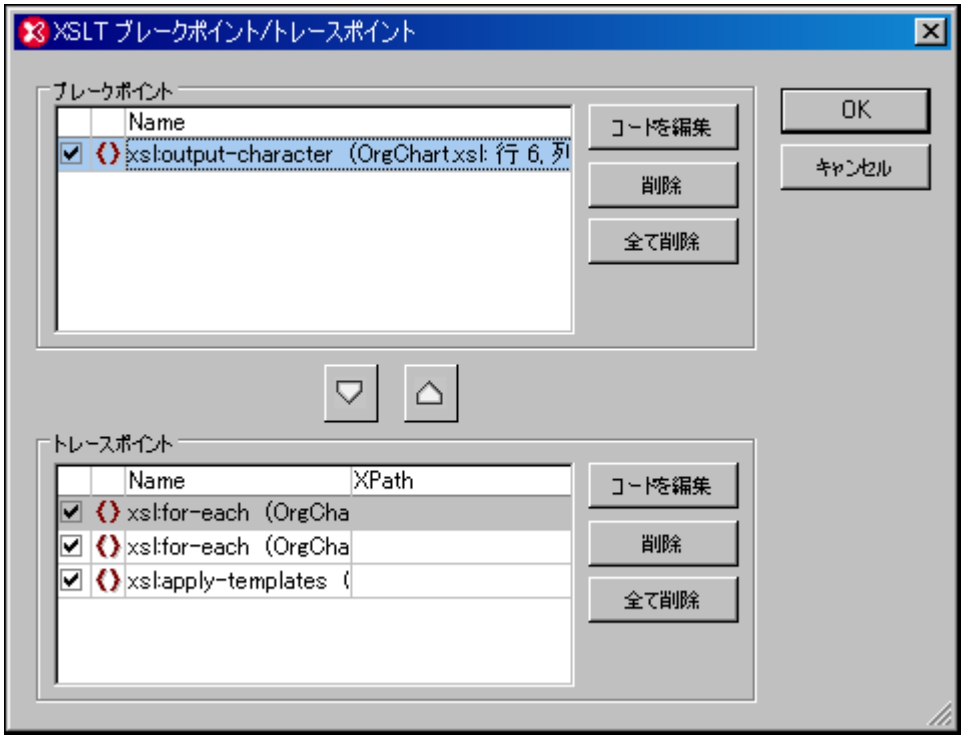
このコマンド(ツールバーアイコンにはありません)により、既に定義されたトレースポイントを有効/無効化することができます。トレースポイントが無効化されると、青色のトレースポイントが灰色で表示されます。無効化されたトレースポイントにおいて出力は行われません。ノード名までカーソルを移動させ、トレースポイントの有効/無効化コマンドをクリックして、トレースポイントを有効/無効化することができます。このコマンドはトレースポイントのある位置で右クリックすることでも使用可能です。

デバッグセッションの終了

デバッグセッションを終了し、デバッグセッションを開始する直前の、通常のXMLSpy ビューに戻ります。デバッグセッションで開かれた出力ドキュメントを開かれたままにするかは、XSLT/XQuery デバッガー設定ダイアログ 362 で設定することができます。

XSLT ブレークポイント/トレースポイントダイアログ

このコマンドによりXSLT/XQuery ブレークポイントならびにトレースポイントダイアログが開かれ、デバッグセッションにあるファイルで現在定義されている(無効化されているものも含めて)全てのブレークポイント/トレースポイントが表示されます。

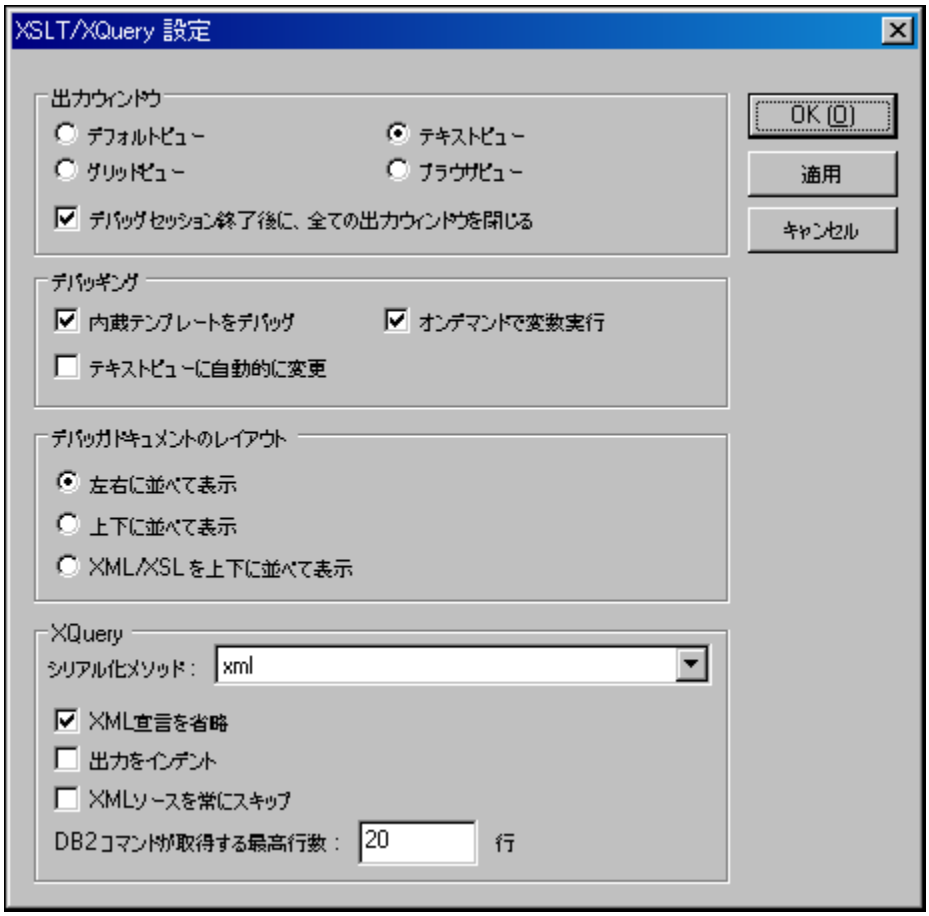


チェックボックスはブレークポイント/トレースポイントが有効になっているか(チェックされていれば有効)を表します。ブレークポイント/トレースポイントを選択し、「削除」ボタンを押すことで、ブレークポイント/トレースポイントの削除を行え、「全て削除」ボタンを押すことで、全てのブレー

クポイントまたはトレースポイントを削除することができます。「コードを編集」ボタンにより選択されたブレークポイント/トレースポイントへ移動します。

9.1.3 デバッガー設定

XSLT and XQuery Debugger 設定ダイアログでは、全てのデバッグセッションに適用されるデバッグならびに出力オプションを設定することができます。設定ダイアログを開くには、「XSL/XQuery | XSLT/XQuery の設定」を選択するか、ツールバーにある アイコンをクリックしてください。このセクションでは設定の詳細について記述します。



出力ウィンドウ

出力ドキュメントウィンドウを(デフォルト、テキスト、グリッド、またはブラウザービューから)指定します。デフォルトビューとはオプションダイアログ(「ツール| オプション 1014」)のファイルの型のセクションにある(.xslt や .xq といった)ファイルの型にて選択されているビューのことです。XSLT 変換では、出力ファイルの型はXSLT ファイルにて定義されます。XQuery の実行では、出力ファイルの種類は、このダイアログのXQuery 設定で指定するシリアル化メソッドにより決定されます(下記参照)。

全ての出力ウィンドウを閉じるオプションにより、デバッグセッション中に開かれていた出力ドキュメントを、デバッグセッションが終了した後にも開いたままにしておくか指定することができます。

デバッグ

内蔵テンプレートをデバッグ設定により、適切な箇所で内蔵テンプレート内のコードに、「**ステップイン**」することができるようになります。これはテンプレートタブの中にあるテンプレートエントリのクリックや、コールスタックにより「**表示**」される内蔵テンプレートファイルとは関係ありません。

XSLT はテキストビューまたはグリッドビューでのみ動作します。テキストビューに自動的に更新するオプションにチェックを入れておくと、デバッグを行うドキュメントがテキストビューやグリッドビューで表示されていない場合に、そのビューが自動的にテキストビューに切り替わります（XQuery デバッガーはテキストビューでのみ動作します）。オンデマンド変数実行チェックボックスがチェックされていれば、変数が呼び出されたときに変数の定義にステップインすることができるようになります。このオプションにチェックが入れられていない場合、変数の呼び出し場所に処理が到達しても、変数の定義へのステップインは行われず、次のステップへ移動します。

デバッガードキュメントのレイアウト

デバッガードキュメントとはデバッガーで開かれているドキュメントのことです。ここではこれらドキュメントを、左右、上下、またはXML/XSLTドキュメントを上下に並べて配置し、結果ドキュメントが左右になるような配置をすることができます。

9.1.4　デバッグセッションの開始

デバッグセッションを開始する最も単純な方法は、単一のXSLT、XQuery、またはXML ファイルから開始するというものです。必要な関連ファイル（関連するファイルのテーブル 356 を参照）が既にアクティブなファイルに関連付けられている場合、デバッグセッションがすぐに開始されます。そうでない場合は必要な関連ファイルを選択するよう促されます。XQuery ファイルはXML ファイルの割り当てを含まず、必要ともしないので、XQuery ファイルからXQuery デバッグセッションを開始するたびにオプションのXML ファイル割り当てを促さないように設定することができます。

定義済みの割り当て

定義済みの割り当てはXSLT デバッグセッションでのみ有効になるもので、アクティブなファイルに対して既に割り当てファイルが存在する場合に参照されます。XML またはXSLT ファイルにて割り当てを行うには、以下の操作を行ってください:

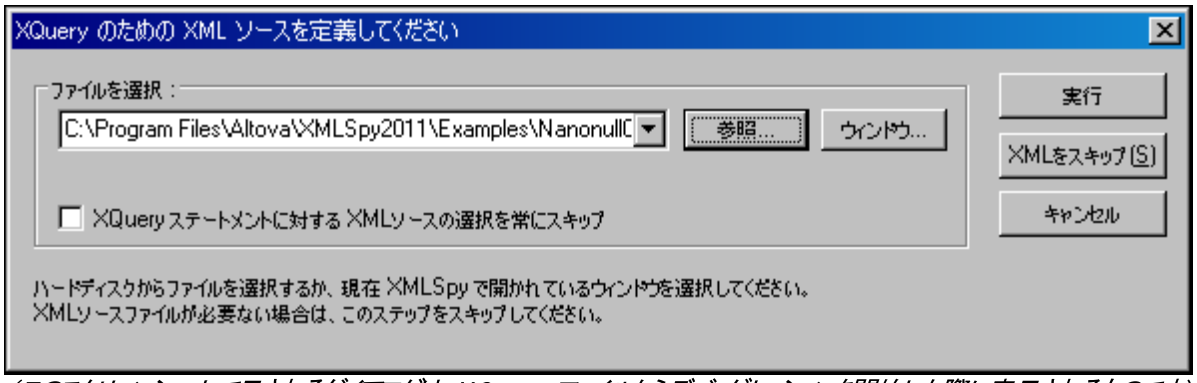
- XML/JSON ファイル: ファイルを開き、「**XSL/XQuery | XSL の割り当て**」からXSLT ファイルを選択します。
- XSLT ファイル: ファイルを開き、「**XSL/XQuery | サンプルXML ファイルの割り当て...**」からXML ファイルを選択します。

「**XSL/XQuery | デバッガーの開始/実行**」を選択すると、デバッグセッションが直接開始されます。アクティブなファイルに対してファイルの割り当てを行うよう促されることはありません。

直接の割り当て

アクティブなファイルに対して定義済みの割り当てが存在しない場合、割り当てを行うよう促されます。「**XSL/XQuery | デバッガーの開始/実行**」を選択すると、以下のような動作が発生します:

- XML ファイルの場合：XSLT またはXQuery、XQuery Update ファイルを選択するよう促されます。
- JSON ファイルの場合： XSLT またはXQuery、XQuery Update ファイルを選択するよう促されます。
- XSLT ファイルの場合：XML ファイルを選択するよう促されます。
- XQuery ファイルの場合：XML ファイルを選択するオプションが与えられます。スキップすることもできます。



（このスクリーンショットで示されるダイアログは、XQuery ファイルからデバッグセッションを開始した際に表示されるものです）

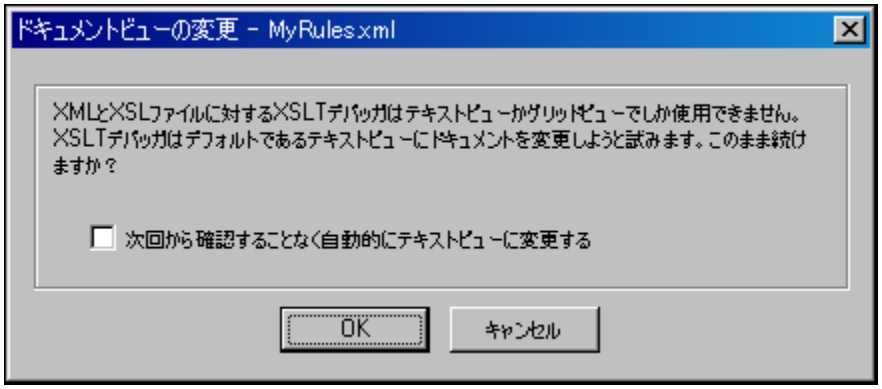
必要な割り当てファイルを選択するか、オプションの割り当てをスキップすると、デバッグセッションが開始されます。

その他のファイル割り当てを行う方法

プロジェクトプロパティダイアログでは、定義済みの割り当てを行うことができます。「**プロジェクト | プロパティ**」を選択し、「**使用する XSL / 使用する XML**」チェックボックスで必要なファイルを割り当ててください。

デバッガービュー

XSLT and XQuery Debuggerはテキストビューおよび拡張グリッドビューでのみ動作します。お使いの XML または XSLT ファイルが、テキストビューまたはグリッドビュー以外のビューで開かれている場合、または SPS ファイルが XML ファイルに割り当てられている場合、デバッグセッションの起動時に以下のダイアログが表示されます。



「**OK**」をクリックすることで、ドキュメントがテキストビューで開かれます。XQuery ファイルは常にテキストビューで表示されることに注意してください。

9.1.5 情報ウィンドウ

デバッグセッション中、XSLT and XQuery Debuggerのインターフェースに表示される情報ウィンドウには XSLT 変換または XQuery 実行に関する様々な情報が表示されます。この情報を XSLT や XQuery ファイルのデバッグを行う際に利用することができます。

(C) 2015-2021 Altova GmbH

XSLT デバッグセッションには8つの情報ウィンドウがあり、XQuery デバッグセッションには5つの情報ウィンドウが与えられます。これらウィンドウはデフォルトで2つのグループに分けられ、XSLT and XQuery Debuggerインターフェースの下部に配置されます（以下の図を参照）。このセクションでは、これらウィンドウと表示される情報に関する説明を行ないます。

XSLT and XQuery Debuggerインターフェースのデフォルトレイアウト。

情報ウィンドウにある最初のグループは、以下のタブとして単一のウィンドウに表示されます。

- コンテキスト 365 （XSLT デバッグセッションのみ）
- 変数 366
- XPath-Watch 367

情報ウィンドウにある第2のグループは、以下のタブとして単一のウィンドウに表示されます。

- コールスタック 367
- メッセージ 368
- テンプレート 369 （XSLT デバッグセッションのみ）
- Info 369
- トレース 370

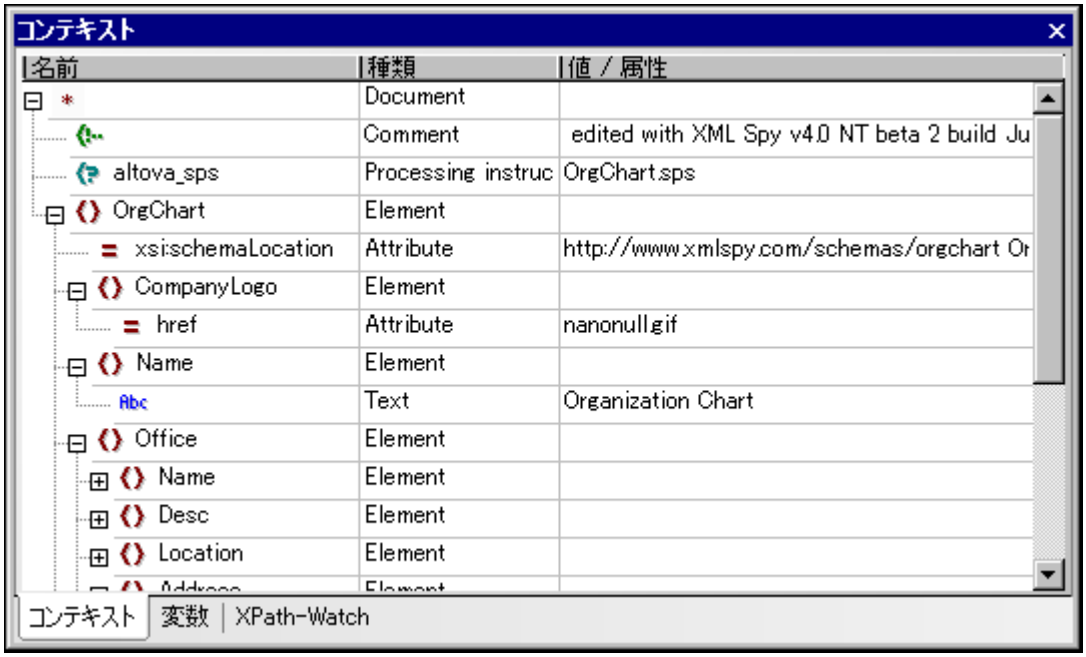
デフォルトのレイアウトでは2つのウィンドウグループが存在し、各ウィンドウには様々なタブが含まれています。同時にアクティブにすることのできるタブは1つのみです。例えば、最初のウィンドウグループにて変数に関する情報を確認するには「**変数**」タブをクリックします。これにより変数情報ウィンドウが表示され、コンテキストならびにXPath-Watch 情報ウィンドウが隠されることになります。タブによってはナビゲーションを行うためのツールとしても使えることに注目してください。アイテムをクリックすることで、XML、XSLT、またはXQuery ファイル内の対応するアイテムに移動することができます。詳細は各情報ウィンドウのドキュメント（コンテキスト 365、コールスタック 367、テンプレート 369）を参照ください。

これら2つの情報ウィンドウグループは、ウィンドウの境界線をドラッグすることで大きさを変更することができます。タブ名をクリックして、ウィンドウの外にドラッグすることで、各情報ウィンドウを切り離すこともできます。ウィンドウのタイトルバーを、ウィンドウグループのタイトルバーにドラッグすることで、グループに加えることもできます。変更したレイアウトをデフォルトのレイアウトにリセットする方法は無いことに注意してください。

9.1.5.1 コンテキストウィンドウ

コンテキストウィンドウはXSLT デバッグセッションでしか利用できません。XQuery デバッグセッションでは表示されません。

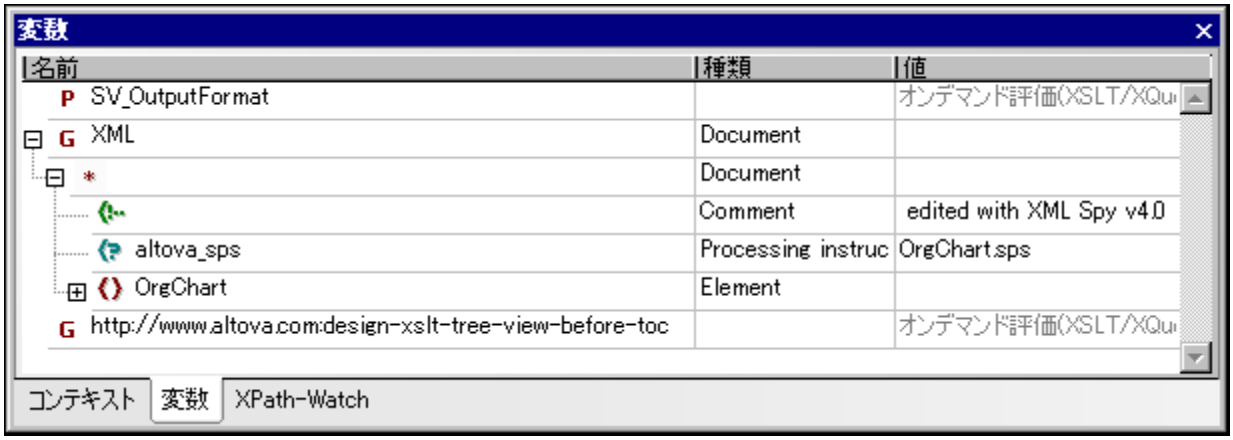
XSLT スタイルシートの処理中、プロセッサーのコンテキストは、常に(ノードやアトム値の)シーケンスにマッチするテンプレート内にあります。コンテキストウィンドウには、現在処理中のコンテキストである、ノード、シングルノード、または(文字列のような)アトム値のシーケンスが表示されます。コンテキストアイテムの種類に従い、値や属性が表示されます。例えばコンテキストアイテムが要素であれば、その要素の属性が表示され、属性またはテキストノードであれば、その値が表示されます。



コンテキストウィンドウのエントリをクリックすることで、XML ドキュメント内にあるそのアイテムが表示されます。XML ドキュメントがインターフェースに表示されていない場合、その XML を表示するウィンドウが開かれます。

9.1.5.2 変数ウィンドウ

変数ウィンドウは XSLT ならびに XQuery デバッグセッションの両方で利用することができます。XSLT/XQuery ドキュメントのスコープ内で使用される変数やパラメーターが、値と共に表示されます。

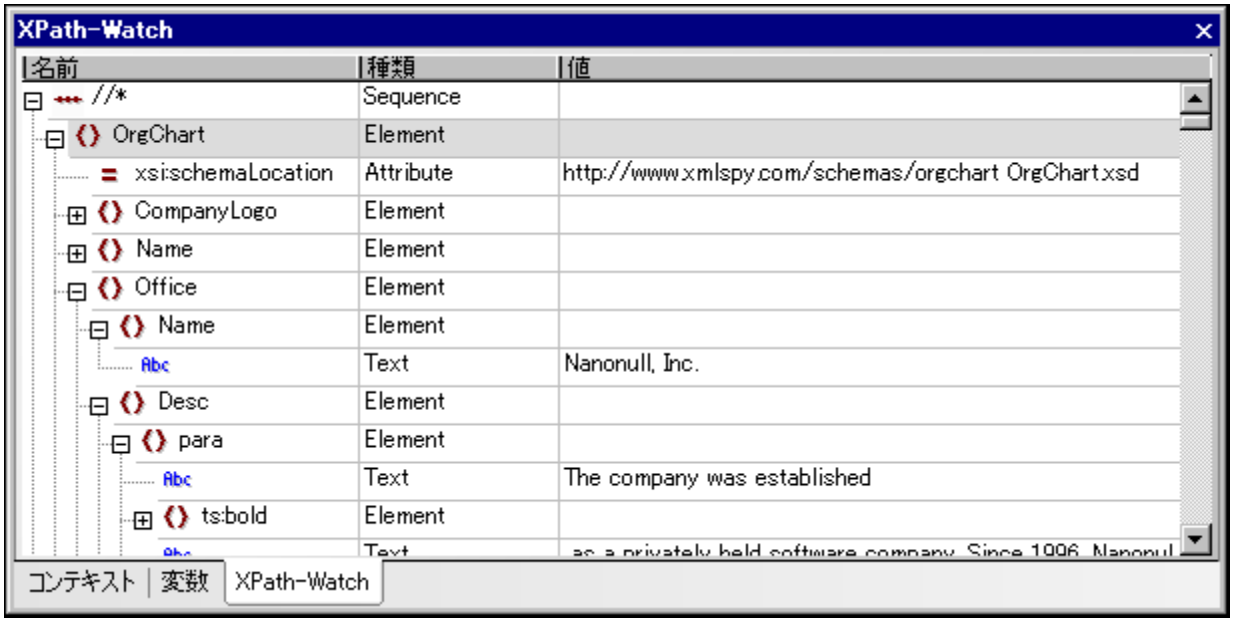


パラメーターは P と共に表示され、(スタイルシートのトップレベルで宣言される) グローバル変数は G と共に表示されます。(XSLT テンプレート内で宣言される) ローカル変数は L で表示されます。変数やパラメーターの値の種類は値フィールドのアイコンにより表示されます。以下の型が区別されます: ノードセット、ノードフラグメント、文字列、数値、Boolean。

9.1.5.3 XPath-Watch ウィンドウ

XPath-Watch ウィンドウは XSLT ならびに XQuery デバッグセッションにて利用可能です。

このウィンドウにより、コンテキスト上で評価を行ないたい XPath 条件式の入力を行うことができます。XSLT ドキュメントのステップを進めていくうちに、その時点のコンテキストで XPath 条件式が評価され、値列に結果が表示されます。



XPath 条件式を入力するには、名前列のテキストフィールドをダブルクリックして、XPath を入力します。他にも、ファイルから XPath 条件式をドラッグして、XPath-Watch ウィンドウにドロップすることもできます。XSLT スタイルシートの XSLT バージョンに合致する XPath のバージョンで使用される条件式を使用してください (XSLT 1.0 では XPath 1.0 が使用され、XSLT 2.0 では XPath 2.0 が使用され、XSLT 3.0 では XPath 3.0 が使用されます)。

注意: XML ファイルまたは XSLT ファイルにて名前空間が使用されている場合、XPath 条件式でも、適切な名前空間プレフィックスを使用する必要があります。

9.1.5.4 コールスタックウィンドウ

コールスタックウィンドウは XSLT ならびに XQuery 両方のデバッグセッションにて利用可能です。

コールスタックウィンドウには、それまで処理した XSLT テンプレートと命令が、リストの最上部にある現在のテンプレートと共に表示されます。

ウィンドウのエントリをクリックすることで、選択されたXSLT テンプレートや命令がXSLT ドキュメントウィンドウ内に表示されます。内蔵のテンプレートを参照するテンプレートや命令をクリックすると、全ての内蔵関数を表示する新たなウィンドウが表示され、選択されたテンプレートがハイライトされます。

9.1.5.5 メッセージウィンドウ

メッセージウィンドウはXSLT とXQuery 両方のデバッグセッションで利用することができます。

XSLT 1.0、XSLT 2.0、およびXSLT 3.0

XSLT デバッグセッションのメッセージタブには、エラー、`xsl:message` 命令、またはデバッグ中に発生したエラーメッセージが表示されます。

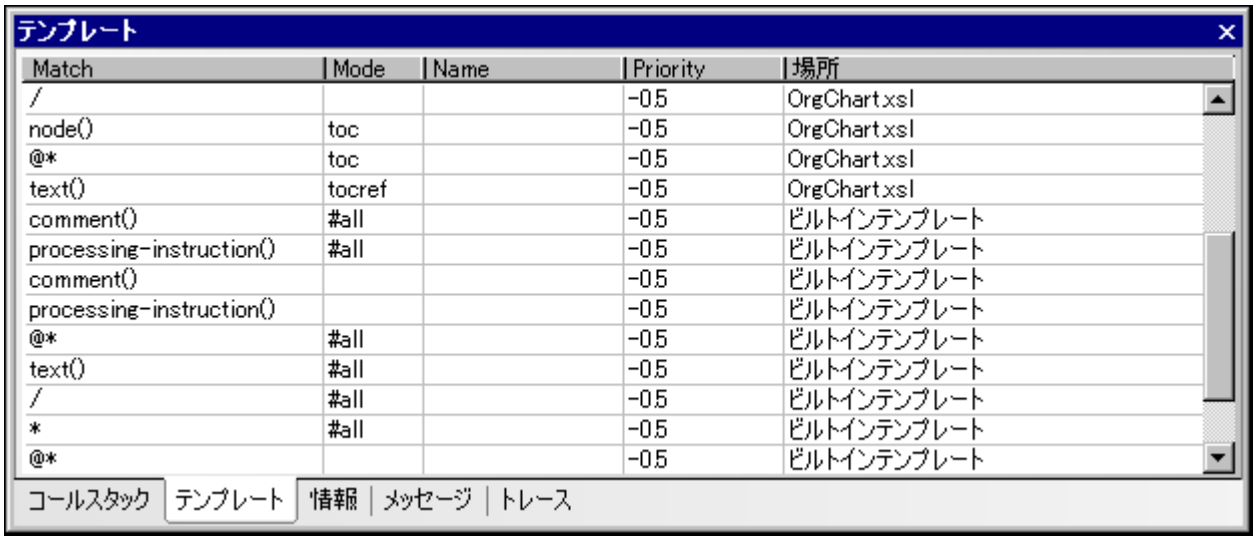
XQuery

XQuery デバッグセッションでは、メッセージウィンドウにエラーメッセージが表示されます。

9.1.5.6 テンプレートウィンドウ

(以下のスクリーンショットにある) テンプレートウィンドウはXSLT デバッグセッションでのみ表示され、XSLT デバッグセッションでは表示されません。

テンプレートウィンドウには、内蔵のテンプレートや名前付きテンプレートなどの、様々な XSLT スタイルシートが表示されます。マッチしたテンプレートは、ノードごとにまとめられて表示されます。名前付きテンプレートは名前順に表示されます。両方の種類のテンプレートにおいて、テンプレートの名前、モード、優先度、そして場所が表示されます。



Match	Mode	Name	Priority	場所
/			-0.5	OrgChart.xsl
node()	toc		-0.5	OrgChart.xsl
@*	toc		-0.5	OrgChart.xsl
text()	tocref		-0.5	OrgChart.xsl
comment()	#all		-0.5	ビルトインテンプレート
processing-instruction()	#all		-0.5	ビルトインテンプレート
comment()			-0.5	ビルトインテンプレート
processing-instruction()			-0.5	ビルトインテンプレート
@*	#all		-0.5	ビルトインテンプレート
text()	#all		-0.5	ビルトインテンプレート
/	#all		-0.5	ビルトインテンプレート
*	#all		-0.5	ビルトインテンプレート
@*			-0.5	ビルトインテンプレート

上のスクリーンショットではXSLT スタイルシートにある3 つのテンプレートが表示されています: ドキュメントノード / にマッチするテンプレート、やテンプレート `n1:italic` と`n1:bold` ノードにマッチするテンプレートです。その他のテンプレートは(場所のフィールドにて示されている通り) 内蔵されている(ビルトインの) テンプレートです。

このウィンドウのエントリをクリックすることで、XSLT ドキュメントウィンドウにあるテンプレートがハイライトされます。内蔵のテンプレートをクリックすると、全ての内蔵テンプレートが表示されるウィンドウが新たに開かれ、選択したテンプレートがハイライトされます。

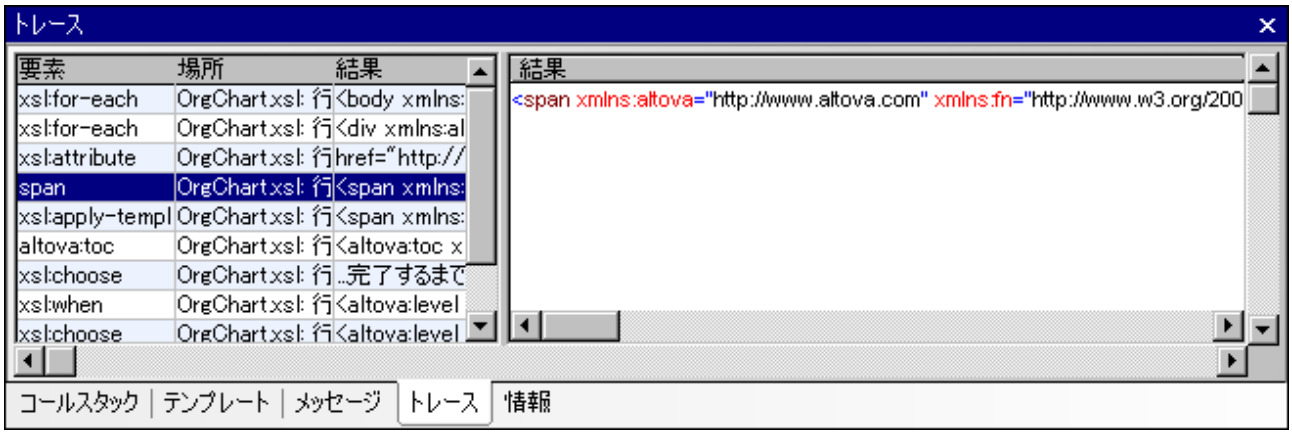
9.1.5.7 情報ウィンドウ

情報ウィンドウはXSLT とXQuery 両方のデバッグセッションで使用することができます。現在のデバッグセッションに関するメタ情報が表示されます。この情報の中には使用されているデバッガーや、ソースならびに出力ドキュメント、そしてデバッガーのステータスなどが表示されます。

デバッガの種類	XSLT 2.0 デバッガ
XML ドキュメント	C:¥Program Files¥Altova¥XMLSpy2011¥Examples¥OrgChart.xml
XSL ドキュメント	C:¥Program Files¥Altova¥XMLSpy2011¥Examples¥OrgChart.xsl
出力ドキュメント	XSL Output.html
デバッガのステータス	デバッガ停止中

9.1.5.8 トレースウィンドウ

トレースウィンドウはXSLT とXQuery 両方のデバッグセッションで利用することができます。



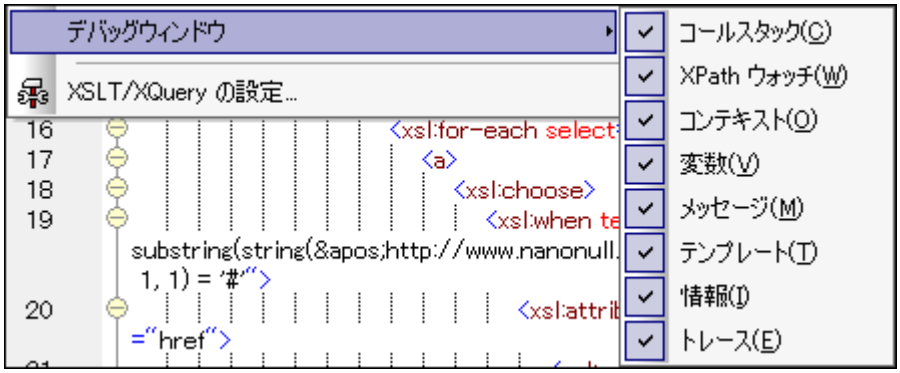
トレースウィンドウにはトレースポイントがセットされた要素、XSLT スタイルシート上の場所、そして要素が実行された際に生成される結果が表示されます。ウィンドウの左側にある行をクリックすることで、完全な結果が右側に表示されます。

9.1.5.9 情報ウィンドウの配置

情報ウィンドウをXSLT and XQuery Debuggerインターフェースの内部に配置することもできます。ウィンドウはインターフェースにドッキングさせることも、固定させないでおくことも、更に別のウィンドウにて1つのペインとして配置することも可能です。以下のメカニズムを利用して、ウィンドウの配置を行うことができます。

メニュー

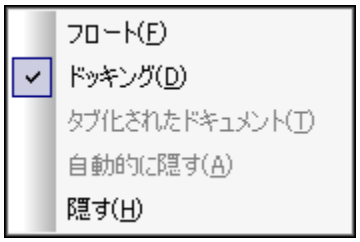
「XSL/XQuery」メニューにて、カーソルを「デバッグウィンドウ」に移動すると、情報ウィンドウのリストが表示されます。メニューアイテムをクリックすることでウィンドウの表示、非表示を切り替えることができます。



ウィンドウ表示のオン、オフを切り替えることができます。

コンテキストメニュー

ウィンドウタブ、または、タイトルバーを右クリックすることによりコンテキストメニューにアクセスすることができます。



目的のオプションをクリックすることで、ウィンドウを切り離し、ドッキングすることができます。

ドラッグアンドドロップ

タブ、または、タイトルバーからウィンドウをドラッグし、希望する場所に配置することができます。

更に、ウィンドウを他のウィンドウ、または、インターフェイスにウィンドウをドラッグすると表示される配置コントロールを使用してドックすることができます：

- ウィンドウを他のウィンドウ上にドラッグすると、配置コントロールが表示されます(*してのスクリーンショット参照*)。ウィンドウを別のウィンドウにドラッグすると、(以下のスクリーンショットにある)丸型の配置コントロールが表示されます。このコントロールは5 つの部分に分かれており、これらのうちのどれかでマウスボタンを放すと、ドラッグしていたウィンドウが「**ターゲットとなるウィンドウ**」の対応する場所に配置されます。端にある4 つのエリアにより、ターゲットとなるウィンドウの、対応する側への配置を行うことができます。中央のボタンは、ドラッグされたウィンドウをターゲットウィンドウのタブとしてドックします。タブバーにドラッグしてドロップすることにより、他のウィンドウ内のタブとしてウィンドウをドックすることもできます。

- ウィンドウをドラッグすると、4 つの矢印からなる配置コントロールが表れます。各矢印はエディター、または、**デバッガーインターフェイス**のサイドにそれぞれ対応しています。ドラッグされたウィンドウをこれらの矢印の一つの上でリリースすると、ドラッグされたウィンドウがエディター、または、デバッガーインターフェイスのサイドにドックされます。

ウィンドウのタイトルバーをダブルクリックすることでも、ウィンドウのドッキング、非ドッキング状態を切り替えることができます。

9.1.6 ブレークポイント

XSLT and XQuery Debugger ではXSLT、XQuery、そしてXMLドキュメントにてブレークポイントを定義することができます。ブレークポイントは(以下のスクリーンショットにあるように)赤色の点線で表示されます。

メモ: 同一の場所にブレークポイントとトレースポイントの両方をセットすることも可能です。(スクリーンショットにあるように)青と赤による線として表示されます。

```
<!– Local Variable reused –>
<xsl:variable name="OfficeName" select="n1:Name"/>
<!– Display the company name if the varaible is true ********* –>
<xsl:if test="$Show_Company_Name">
    <h3>
        <xsl:value-of select="$OfficeName"/>
    </h3>
    <xsl:message>
        <xsl:text>Company Named Displayed</xsl:text>
    </xsl:message>
</xsl:if>
```

デバッグセッションでデバッガーを開始すると、各ブレークポイントにてデバッギングが停止します。この方法を使うと、デバッグを行う場所を限定し、XSLT、XQuery、そしてXMLドキュメント内の特定の場所に注意を集中させることができます。セットできるブレークポイントの数に制限はありません。

注意:ドキュメント内にセットされたブレークポイントの情報は、そのドキュメントが閉じられるまで保持されたままになります。しかし(例えばXSDドキュメントにて)スキーマビューへの切り替えが行われると、ブレークポイントは削除されます。その後テキストビューやグリッドビューに切り替え直しても、ブレークポイントは表示されません。

XMLドキュメント内のブレークポイント

XMLドキュメントでは任意のノードにブレークポイントをセットすることができます。ブレークポイントによる処理の停止は、そのノードの開始点で行われます。

XSLTドキュメント内のブレークポイント

XSLTドキュメントでは以下の箇所でブレークポイントをセットすることができます:

- テンプレートや(例えば`xsl:for-each`といった)テンプレート命令の開始箇所
- XPath 条件式(XPath 1.0 またはXPath 2.0)
- 文字列により構成されるXML フラグメント。ブレークポイントによる処理の停止は、そのノードの開始点で行われます。

XQueryドキュメント内のブレークポイント

XQuery ドキュメントでは、以下の箇所でブレークポイントをセットすることができます:

- XQuery ステートメントの開始箇所
- XQuery 表現内部
- 文字列により構成されるXML フラグメント。ブレークポイントによる処理の停止は、そのノードの開始点で行われます。

ブレークポイントの挿入と削除

ブレークポイントを挿入するには:

1. ドキュメント内でブレークポイントを挿入したい箇所にカーソルを移動させます(上の記述を参照ください)。XSLT デバッグセッションでは、テキストビューならびにグリッドビューにてブレークポイントをセットすることができ、XQuery デバッグセッションでは、テキストビューでのみブレークポイントを使用することができます。
2. 以下のどれかの操作を行います:
   - 「XSL/XQuery | **ブレークポイントの挿入/削除**」を選択
   - **F9** を押下
   - 右クリックして、「**ブレークポイントの挿入/削除**」を選択

ブレークポイントを削除するには:

1. ドキュメント内のブレークポイントが存在する箇所にカーソルを移動させます。
2. 以下のどれかの操作を行ないます:
   - 「XSL/XQuery | **ブレークポイントの挿入/削除**」を選択
   - 「F9」を押下
   - 右クリックして「**ブレークポイントの挿入/削除**」を選択

その他にも、ブレークポイント/トレースポイントダイアログを使用することでブレークポイントを削除することができます:

1. メニューオプションから「XSL/XQuery | **ブレークポイント/トレースポイント**...」を選択
2. ダイアログボックスからブレークポイントを選択して、「**削除**」ボタンをクリック

「**全て削除**」ボタンにより、ダイアログボックス(そして XSLT スタイルシートにある)全てのブレークポイントを削除することができます。

ブレークポイントの有効化/無効化

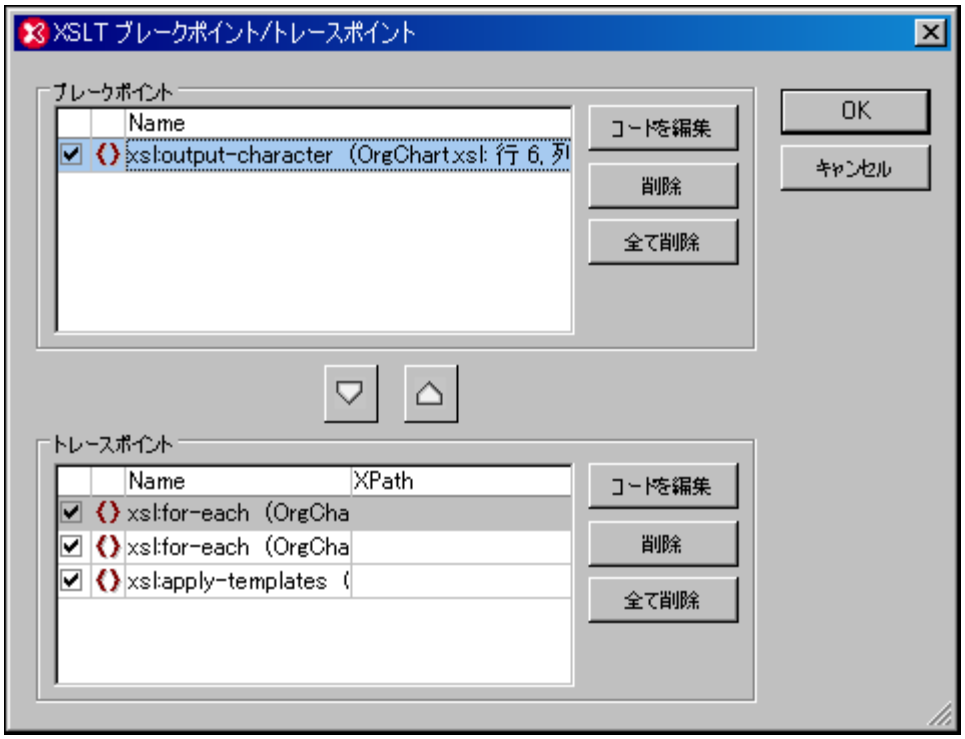
ブレークポイントを挿入した後に、そのブレークポイントを(削除すること無く)スキップしたい場合は、ブレークポイントの無効化を行うことができます。必要に応じて、ブレークポイントは再度有効化することができます。

ブレークポイントを無効化するには:

1. ブレークポイントが含まれているノードまたは表現にカーソルを移動します。
2. 「XSL/XQuery | **ブレークポイントの有効化/無効化**」を選択します(または「Ctrl」+「F9」を押下します)。ブレークポイントが赤色からグレーになり、無効化されます。

その他にも、ブレークポイント/トレースポイントダイアログを使用することでブレークポイントの無効化を行うことができます:

1. メニューオプションから「XSL/XQuery | **ブレークポイント/トレースポイント**...」を選択して、XSLT ブレークポイント/トレースポイントダイアログを表示します。ダイアログには開かれている XML ソースならびに XSLT スタイルシートにて現在定義されている全てのブレークポイントが表示されます。



2. 無効化したいブレークポイントのチェックマークを外し、「OK」ボタンにて確定します。ブレークポイントが赤色からグレーになり無効化されます。

ブレークポイントを有効化するには:

1. ブレークポイントを含んだコードや表現にカーソルを移動します。
2. 「XSL/XQuery | ブレークポイントの有効化/無効化」を選択します(または「Ctrl」+「F9」を押下します)。ブレークポイントがグレーから赤色になり、有効化されます。

特定のブレークポイントを見つける

特定のブレークポイントを見つけるには:

1. メニューオプションから「XSL/XQuery | ブレークポイント/トレースポイント...」を選択します。XSLT ブレークポイント/トレースポイントダイアログが表示されます。
2. ブレークポイントのリストから目的のブレークポイントをクリックします。
3. 「コードを編集」ボタンをクリックします。ブレークポイントダイアログが閉じられ、テキストビューにて選択したブレークポイントの直前にカーソルが配置されます。拡張グリッドビューの場合、ブレークポイントを含んだテーブルセルが赤色でハイライトされます。

ブレークポイントの後でデバッグを継続する

ブレークポイントの後にある処理を継続するには:

- 「XSL/XQuery | ステップイン」または「XSL/XQuery | デバッガーの開始/実行」コマンドを選択してください。

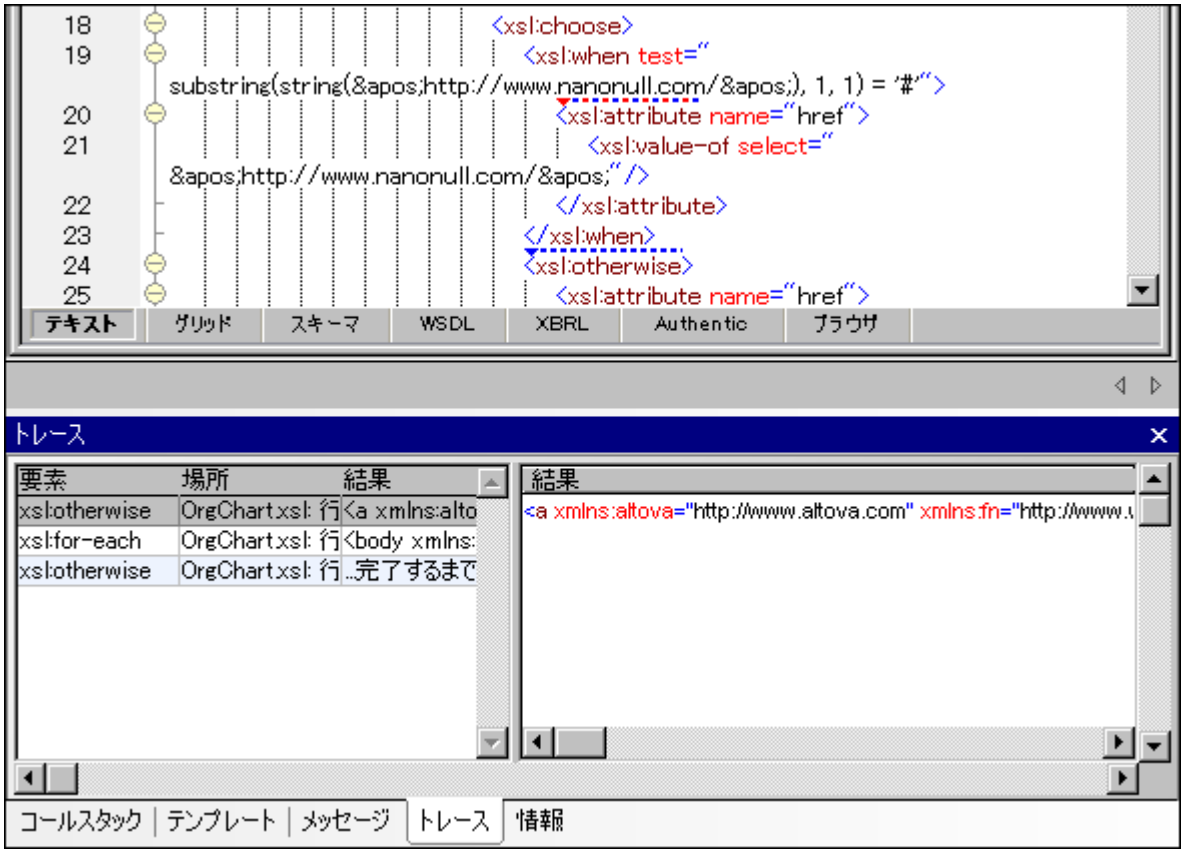
9.1.7 トレースポイント

XSLT and XQuery Debuggerでは XSLT ドキュメントのトレースポイントを定義することができます。

トレースポイントを使うことで、命令により生成されたコンテンツをトレースし、(xsl:message 要素を使ってメッセージを表示させるなどの) XSLT スタイルシートの編集を行うこと無く、トレースポイントがセットされた箇所における XPath 条件式の結果を確認することができます。

トレースポイントは(下のスクリーンショットにあるように) XSLT スタイルシートにて青色の点線で表示されます。

メモ: 同一の場所にブレークポイントとトレースポイントの両方をセットすることも可能です。(スクリーンショットにあるように) 青と赤による線として表示されます。



トレースポイントがセットされた箇所では、各命令により生成された結果が出力されます。この出力はトレースウィンドウで確認することができます。セットすることのできるトレースポイントの数に制限はありません。

注意: セットされたトレースポイントはドキュメントが閉じられるまで保持されます。

XSLT ドキュメントのトレースポイント

トレースポイントは XSL 命令や XSLT スタイルシートのリテラル結果にセットすることができます。

XML と XQuery ドキュメントのトレースポイント

トレースポイントを XML や XQuery ドキュメントにセットすることはできますが、これらのトレースポイントは効果を持ちません。

トレースポイントの挿入/削除

トレースポイントを挿入するには:

1. XSLT ドキュメント内でトレースポイントをセットしたい箇所にカーソルを移動させます。デバッグセッションでは、テキストビューならびにグリッドビューでトレースポイントをセットすることができます。
2. 以下のどれかの操作を行ないます:

   - 「XSL/XQuery | **トレースポイントの挿入/削除**」を選択
   - 「Shift」+「F9」を押下
   - 右クリックして「**トレースポイントの挿入/削除**」を選択

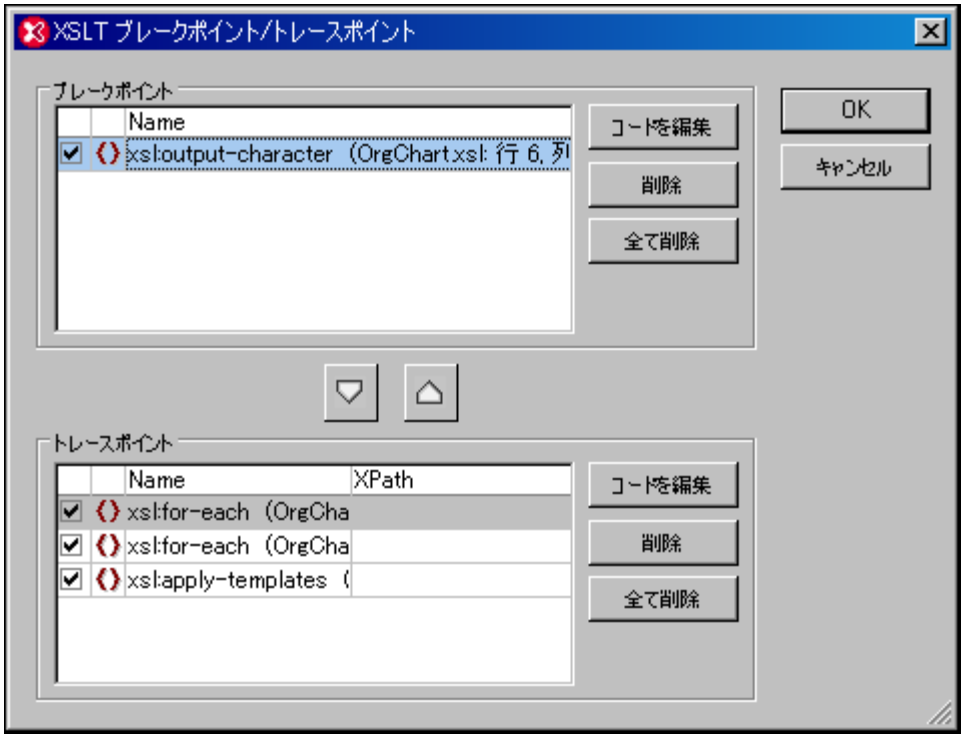
トレースポイントを削除するには:

1. XSLT ドキュメント内のトレースポイントがある場所にカーソルを移動させます。
2. 以下のどれかの操作を行ないます:

   - 「XSL/XQuery | **トレースポイントの挿入/削除**」を選択
   - 「Shift」+「F9」を押下
   - 右クリックして「**トレースポイントの挿入/削除**」を選択

他にも、XSLT ブレークポイント/トレースポイントダイアログからトレースポイントの削除を行うことができます:

1. メニューオプションから「XSL/XQuery | **ブレークポイント/トレースポイント...**」を選択します。
2. (スクリーンショットのように) ダイアログボックスのトレースポイントをクリックし、「**削除**」をクリックします。



トレースポイントペインにある「**全て削除**」ボタンにより、ダイアログボックスに表示されている(そして XSLT スタイルシートにある) 全てのトレースポイントを削除します。

トレースポイントにXPath をセット

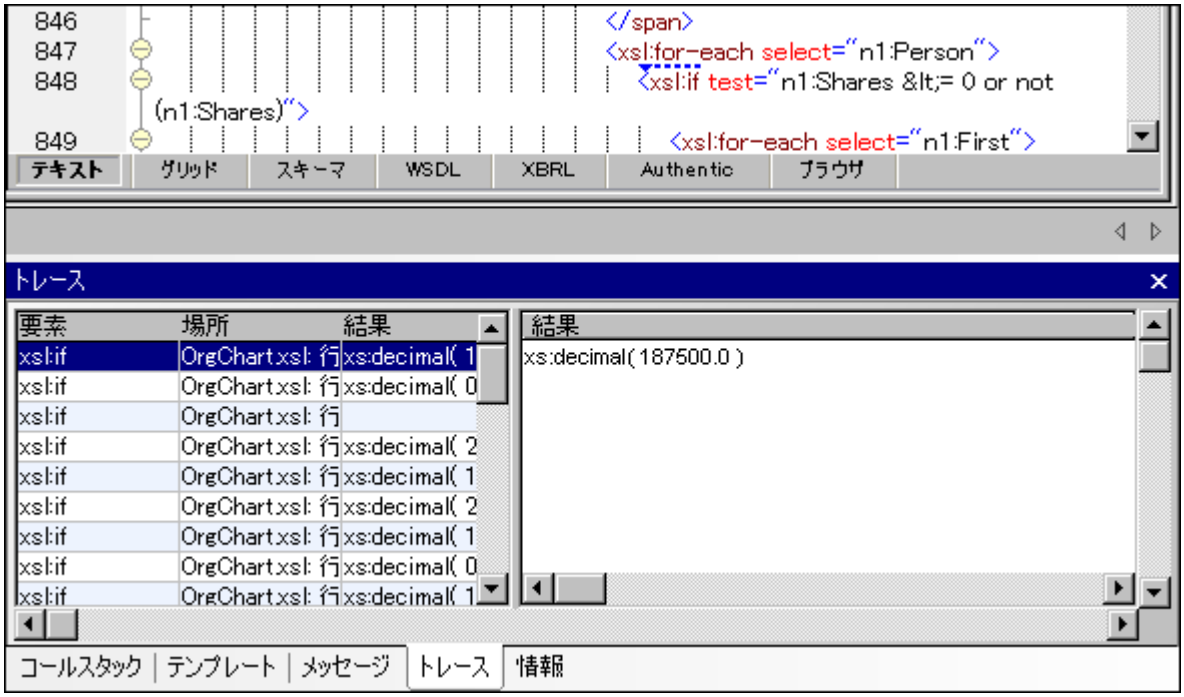
XPath に対してもトレースポイントをセットすることができます。XPath に対してトレースポイントをセットすると、ステートメントにより生成されるコンテンツの代わりに、XPath の評価結果がトレースウィンドウに表示されます。XPath の評価は、トレースポイントがセットされた箇所のコンテキストとは相対的に評価されます。

XPath にトレースポイントをセットするには:

1. メニューオプションから「**XSL/XQuery | ブレークポイント/トレースポイント...**」を選びます。XSLT ブレークポイント/トレースポイントダイアログには、現在開かれている全てのXSLT スタイルシートドキュメント内で定義されているトレースポイントが表示されます。
2. 目的のトレースポイントの行の「**XPath**」フィールドにて XPath を入力します。

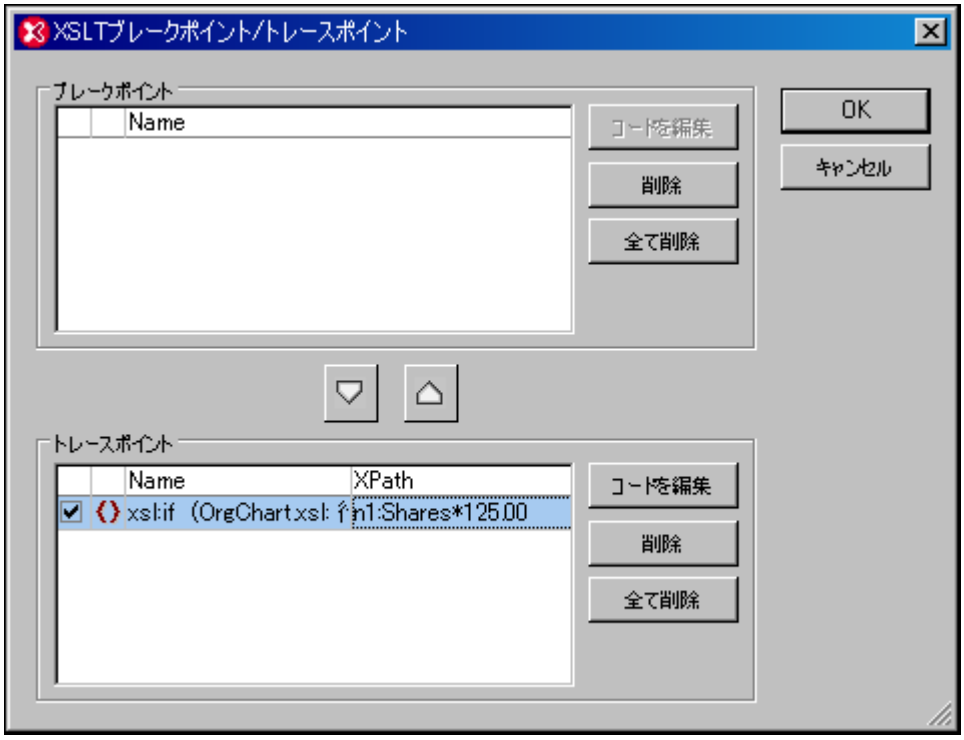
例:

以下のサンプルに示される通り、コンテキストノードが `Person` となるようにトレースポイントがセットされています。`Person` 要素には `Shares` 要素が含まれています。ここでは各 `Person` が持つ `Shares` の数を 125 でかけた値を表示します。



以下の操作を行ってください:

1. 555 行にてトレースポイントをセットします。
2. XSLT ブレークポイント/トレースポイントダイアログを開き、先程セットしたトレースポイントの XPath にて `n1:Shares*125.00` と入力します。

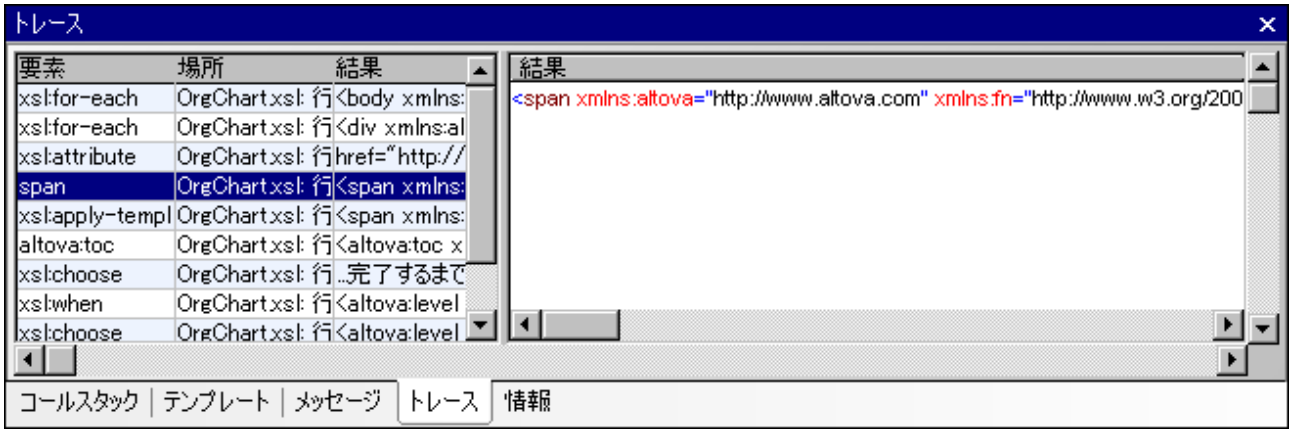


3. デバッガーを開始します。入力した XPath のけかがトレースポイントウィンドウに表示されます。

トレースウィンドウ

「XSL/XQuery | デバッガーの開始/実行」を選択して、デバッグをスタートさせます。トレースポイントがセットされた命令の出力がトレースウィンドウ[370]に表示されます(スクリーンショットを参照)。トレースウィンドウのアイテムをクリックすることで、その結果の詳細がウィンドウの右側に表示されます。

メモ:トレースウィンドウに表示される結果は、トレースされている命令が終了した後に表示されます。



トレースポイントの有効化/無効化

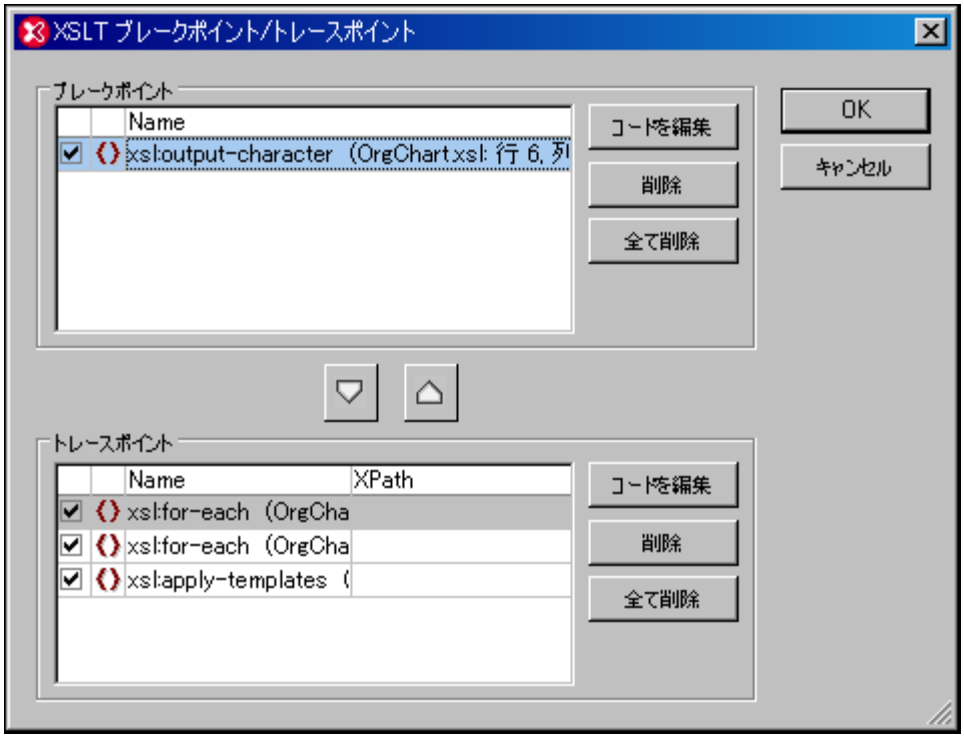
トレースポイントを挿入した後、それらを無効化することで、削除すること無くトレースポイントをスキップすることができます。必要になれば再度有効化することもできます。

トレースポイントを無効化するには:

1. XSLT スタイルシートにある無効化したいトレースポイントにカーソルを移動します。
2. 「XSL/XQuery | トレースポイントの有効化/無効化」を選択するか、「Ctrl」+「Shift」+「F9」を押下します。トレースポイントが青色からグレーに変わり、無効化されます。

その他にも、XSLT ブレークポイント/トレースポイントダイアログからトレースポイントの無効化を行うことができます:

1. メニューオプション「XSL/XQuery | ブレークポイント/トレースポイント...」を選択します。XSLT ブレークポイント/トレースポイントダイアログボックスが開かれ、開かれている XSLT スタイルシートドキュメント内で現在定義されている全てのトレースポイントが表示されます。



2. 各トレースポイントにあるチェックを外すことで、トレースポイントを無効化して、「OK」で確定します。トレースポイントが青色からグレーに変わり、無効化されます。

トレースポイントを有効化するには:

1. XSLT スタイルシートにある有効化したいトレースポイントにカーソルを移動します。
2. 「XSL/XQuery | トレースポイントの有効化/無効化」を選択するか、「Ctrl」+「Shift」+「F9」を押下します。トレースポイントが青色からグレーに変わり、無効化されます。

特定のトレースポイントを見つける

特定のトレースポイントを見つけるには:

1. メニューオプションから「XSL/XQuery | ブレークポイント/トレースポイント...」を選択します。XSLT ブレークポイント/トレースポイントダイアログが表示されます。
2. リストにある目的のトレースポイントをクリックします。
3. 「コードを編集」ボタンをクリックします。XSLT ブレークポイント/トレースポイントダイアログが閉じられ、テキストビューにて選択したトレースポイントの直前にカーソルが配置されます。拡張グリッドビューの場合、トレースポイントを含んだテーブルセルが青色でハイライトされます。

9.1.8 デバッガーショートカット

デフォルトのデバッガーショートカットは以下にリストされているとおりです

デバッガーコマンド

F9	ブレークポイントの挿入/削除
F9 + Shift	トレースポイントの挿入/削除
F9 + CTRL	ブレークポイントの有効化/無効化
F9 + Shift + CTRL	トレースポイントの有効化/無効化
F11	ステップイン
F11 + Shift	ステップアウト
F11 + CTRL	ステップオーバー
F11 + Alt	デバッガーを開始する

(C) 2015-2021 Altova GmbH

10 XPath/XQuery 条件式

XPath とXQuery 式はXML ツリーをナビゲートするために使用されます。バージョン 3.1 内のマップと配列のためのサポートに加え、バージョン 3.1 のXPath とXQuery 式もJSON 構造をナビゲートするために使用することができます。XPath はXQuery のサブセットで、両言語で有効な式は両方の言語で同じ結果を返します。2 つの言語に関する詳細に関してはXPath 3.1 勧告 とXQuery 3.1 勧告 を参照してください。

XMLSpy 内でXPath/XQuery 式を評価する方法

XMLSpy はXPath/XQuery 式を作成し、式をXML とJSON ドキュメントに対して評価しデバッグするためのパワフルな機能を提供します。これにより、使用するXML またはJSON ドキュメントに対して式を素早くビルドしテストすることができます。

出力ウィンドウが他の出力ウィンドウ[103] と共にアプリケーションのインターフェイスの下に表示されている「XPath/XQuery ウィンドウ」内でこれらの分析機能を使用することができます。XPath/XQuery ウィンドウのこれらの機能はこのセクションのサブセクションで説明されています。

一般的なユーザーシナリオでは、以下を行います:

1. 式をビルドまたは評価するXML またはJSON ドキュメントを開きます。
2. XPath/XQuery ウィンドウ内にXPath/XQuery 式を入力します。
3. 結果を確認するためにエバリュエーターまたはデバッガーを実行します。エバリュエーターは最終的な結果を表示しますが、デバッガーは各ステップの結果アスペクトを表示し評価プロセスをステップバイステップに確認します。

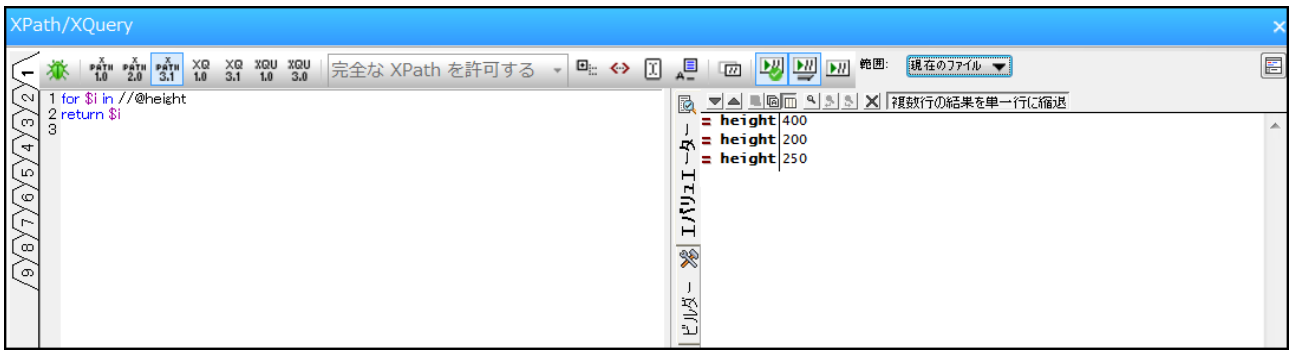
このセクション

このセクションでは以下のように整理されたXPath/XQuery ウィンドウの機能について説明されています:

- Path/XQuery ウィンドウについて[382]
- 評価モード[384]: XPath/XQuery 式の評価のためのモード
- デバッグモード[389]: XPath/XQuery 式のデバッグのためのモード
- 式ビルダー[395]: XPath/XQuery 式の作成のためのビルダー
- JSON のためのXQuery 式[397]
- 注意点[399]

10.1 XPath/XQuery ウィンドウについて

XPath/XQuery ウィンドウ[108] (*下のスクリーンショット*) によりXPath とXQuery 式をXML またはJSON* ドキュメントに対して作成、評価、およびデバッグできるようになりました。



* JSON クエリを有効化する機能はXPath/XQuery 3.1 で導入されました。*XSLT/XQuery を使用した JSON の変換*[541] *を参照してください。*

水平的および垂直的なレイアウト

ツールバーの右角に垂直および水平レイアウトを切り替えるボタンがあります (*上と下のスクリーンショットを参照してください*)。どのモードでもレイアウトを切り替えることができます (*評価モードとデバッグモード*[382] *を参照してください*)。上のスクリーンショットは XPath/XQuery 式が複数のラインで垂直的なレイアウトを表示しています (上のスクリーンショット内の左側のペイン)。

水平的なレイアウト (*下のスクリーンショット*) は結果が水平的な方向に広がるラインを持つ場合役に立ちます。

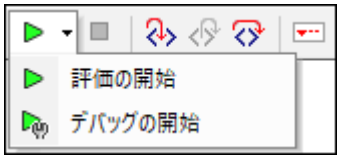
評価モードとデバッグモード

XPath/XQuery ウィンドウ[108] は2 つのモードで使用することができます:

- 1 つまたは複数のXML/JSON ドキュメントに対してXPath またはXQuery 式が評価される評価モード[384]。**式ペイン** に式は入力され、結果は横の「**結果ペイン**」に表示されます。結果内のノードをクリックしてXML またはJSON ドキュメント内のそのノードに移動することができます。
- XPath/XQuery 式を現在アクティブな XML ドキュメントに適用しつつデバッグするデバッグモード[389]。ブレークポイントとトレースポイントを設定することができ、評価をステップバイステップで行うことができます。各ステップで変数のコンテンツを確認することができ、評価の追加アスペクトをチェックするカスタムウォッチ式を設定することができます。

(C) 2015-2021 Altova GmbH

2つのモードを切り替えるには、ウィンドウのツールバーの左角の「**評価/デバッグの開始**」ドロップダウンメニュー内で適切なコマンドを選択します（*下のスクリーンショットを参照*）。



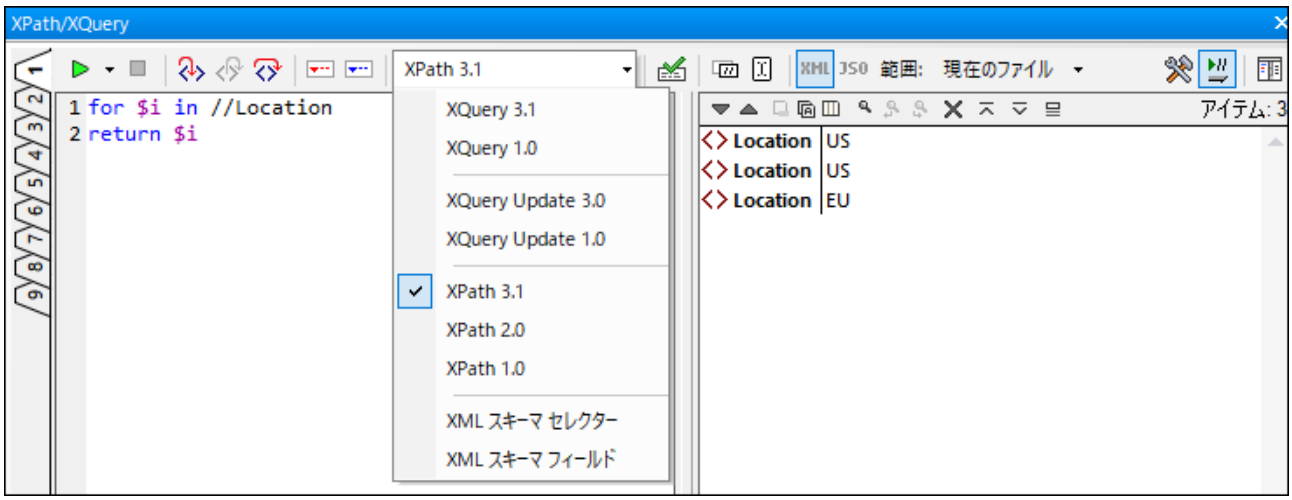
2つのモードの使用方法はこのセクションのサブセクションで説明されています。

XPath/XQuery 式ビルダー

両方のモード 382 内で、式ビルダー 395 を使用してシステム的に正確な式を作成する手助けを行います。Switch 式ビルダー 395 をメインツールバー の「**ビルダーモード**」ボタンを使用してオンとオフに切り替えます。

10.2 式の評価

XPath/XQuery ウィンドウによりXPath またはXQuery 式を選択する言語で 作成することができ、式を選択範囲オプションで指定する範囲で評価することができます（*下のスクリーンショットを参照*）。式は現在のファイル上、および複数のファイルの組み合わせで評価することができます: (i) 現在開かれているドキュメント (ii) 現在アクティブな XMLSpy プロジェクト 676 のファイル(iii) 選択されたフォルダーのファイル。式を単純にテストするには、スコープとして1 つの適切なファイルのみが必要とされます。XPath/XQuery ウィンドウは結果ペイン内の特定の1 つまたは複数のファイルとレポートを検索するために使用することができます。この場合スコープ内で適切なファイルセットを選択してください。

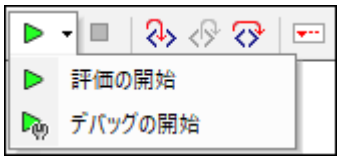


XPath/XQuery ウィンドウはツール バーと式ペイン（*上のスクリーンショットの左側*）と結果ペイン（*上のスクリーンショットの右側*）2 つのペインにより構成されています。

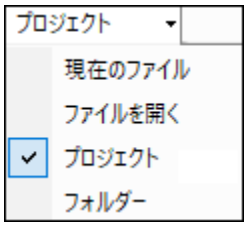
プロシージャの評価

XPath/XQuery 式を評価するには、以下を行います（*上のスクリーンショットを参照してください*）:

1. *評価モードの選択:* ツール バーの左上にある「**評価/デバッグの開始** (F5)」コマンドのプルダウンメニューから「**評価の開始**」を選択します（*下のスクリーンショットを参照*).

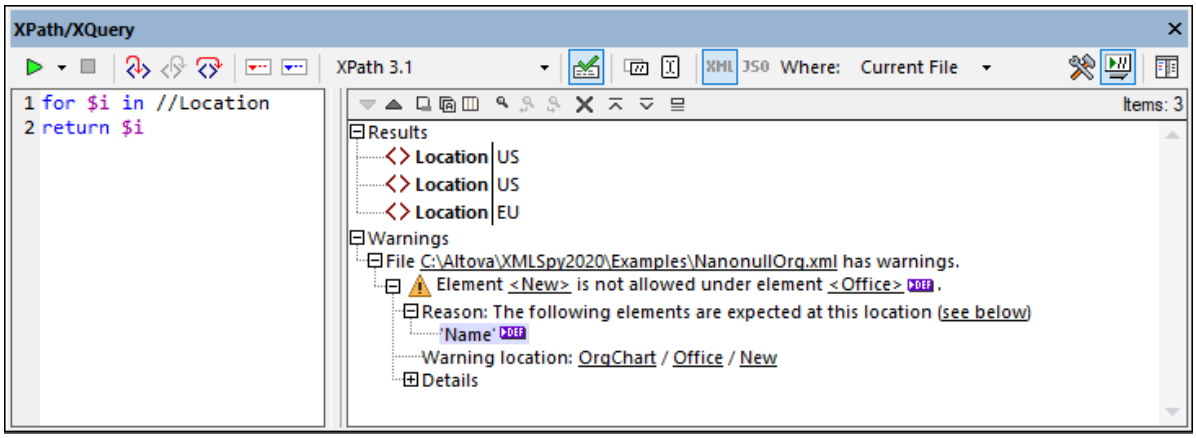


2. *水平的な/垂直的なレイアウト:* レイアウトを切り替えるには、（ツール バーの右側にある）「**水平的な/垂直的なレイアウト**」ボタンをクリックします。デフォルトは垂直的なレイアウトです。
3. *言語バージョンの選択:* ツール バー内で、評価する式のための言語バージョンを選択します。例えば、XPath 3.1 の場合、デフォルトはXPath 3.1 です。以下の *XML スキーマセレクタと XML スキーマフィールド* 386 も参照してください。
4. *式の入力:* 式ペイン内に、評価する式を入力します。式の作成を手助けするために、式ビルダー 395 の入力ヘルパーを使用します。式ペインの編集機能に関する詳細は、下記の*式ペイン内での編集* 386 を参照してください。
5. *評価スコープの選択:* ツール バー内の選択範囲オプション内で、式が評価されるファイルを選択します。オプション: *現在のファイル*、*ファイルを開く*、*（XMLSpy）プロジェクト*、*またはフォルダー*。デフォルトは *現在のファイル*です。



*現在のファイル*が選択されている場合、現在アクティブなファイルが必要になります。*開かれているファイル*を選択すると現在 XMLSpy で開かれているすべてのファイルに対して式が評価されます。プロジェクトは現在アクティブな XMLSpy プロジェクト 676 を指しています。(XMLSpy プロジェクト内の外部フォルダーは*アタ外部フォルダーをスキップ*アイコンをチェックすることにより除外することができます)。*フォルダー*オプションにより必要なフォルダーを参照することができ、XPath 式はこのフォルダー内の XML または JSON ファイルに対して評価されます。

6. *XML または JSON 評価の選択:* 現在のファイルが評価のスコープの場合、評価モード (XML または JSON) は自動的に (JSON-適合ファイルのためにはJSON モードが、非-JSON ファイルのモードためにはXML など)ドキュメントのファイル型の適合型 1018 により決定されます。両方のボタンは有効化され必要な評価モードを選択することができます。デフォルトは以前に選択されていたモードです。
7. *コンテキストノードの設定:* コンテキストノードを以下により設定することができます: (i) ルートノード (ii) アクティブなドキュメント内の現在の選択。 ツールバーボタン「XPath/XQuery のために現在の選択を起点とする」を介して2つの設定間を切り替えることができます。デフォルトの設定はルートノードです。
8. *XML 検証:* XML の評価 ツールバーボタンがオンに切り替えられている場合(デフォルトの設定), 評価されている XML ファイルが評価されます。エラーは結果ペイン内でレポートされる警告として扱われますが評価は継続されます (*下のスクリーンショット*)。



9. *式の評価:* ツールバーの切り替えオプション「入力中に評価」が選択されている場合、結果ペイン内に入力中の評価の結果が表示されます。このオプションが選択されていない場合、評価はツールバーの左上にある)「評価/デバッグの開始 (F5)」をクリックして明示的に開始される必要があります。

評価プロシージャ内で使用されるツールバーボタン

	評価/デバッグの開始 (F5)	評価モードの選択を有効化し、評価を開始します。
	評価/デバッグの停止 (Shift+F5)	評価中に有効化されます。評価が長くかかる場合、または永続的なループで行われる場合に評価を中断する際に役に立ちます。
	水平的な/垂直的なレイアウト	水平的および垂直的レイアウト間の切り替えを行います。

	ビルダーに切り替える	式の作成を助けるコンテキストを考慮した入力ヘルパーを提供する式ビルダーに切り替えます。
	現在の選択範囲のXPath をコピーする	XMLドキュメント内のノードのロケーターパスを式ペイン内のカーソルの最後の場所にコピーします。
	XML/JSON 評価モード (XMLとJSON 評価モード間を切り替えます)	ペアのハイライトされているアイコンはアクティブなオプションです。評価スコープが複数のファイルである場合、両方のアイコンは有効化され、1つを選択することができます。それ以外の場合、評価モードはファイル型に従い自動的に検知されます。他のアイコンは無効化されます。
	現在の選択範囲をコンテキストとして設定する	ルートノードと現在の選択範囲の式コンテキストを切り替えます。
	入力中に評価	式の入力中に式の評価がオンに切り替えられます。
	XML の評価	オンに切り替えられるとターゲット XMLドキュメントが検証されます。

XML スキーマセレクタとXML スキーマフィールド
*XML スキーマセレクタとXML スキーマフィールド*オプションは特定のXPath 1.0 ケースの範囲の狭いサブセットのために使用され、XML スキーマ内で一意な識別制約が定義されている場合に役に立ちます。どちらかのオプションが選択されていると、名前テスト（とワイルドカード *）のみがXPath 式内で許可され、述語とXPath 式は使用できない場合があります。更に、*XML スキーマオプションセレクタ*オプションのために子軸の式のみが許可されます。, *XML スキーマフィールド*オプションに関しては子軸上の式が許可されます。詳細に関しては、W 3C's XML スキーマ: 構造と勧告 を参照してください。

式ペイン内での編集
式ペイン内の式の編集について以下の点に注意してください:

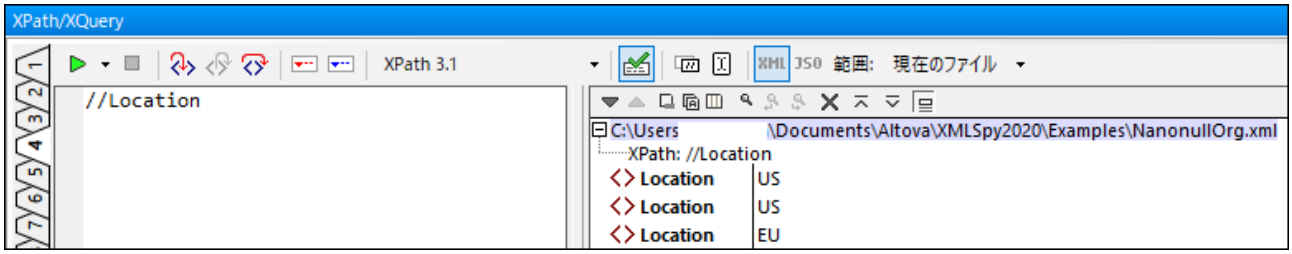
- （わかりやすいように）複数行で式を作成するには、**Return** キーを使用してください。
- 式フィールド内のテキストのサイズを大きくする/小さくするには、式フィールドをクリックし、**Ctrl** を押して、スクロールホイールを回します。*結果ペイン内でこの作用は適用されることに注意してください。*
- ノードのロケータパスを手動で入力する代わりにXMLSpy がパスを入力することができます。これを以下のように行います: (i) XPath 式内でロケータパスを挿入する箇所でカーソルをポイントします。(ii) ターゲットするノードの内部にカーソルをポイントします。(iii) ツールバー内で、ボタン「現在の選択範囲のXPath をコピーする」をクリックします。これにより、選択されたノードのロケーターパスは式内に挿入されます。ロケーターパスはドキュメントのルートノードから開始する絶対パスになります。

XQuery とJSON 評価

- XQuery 評価に関する詳細は、セクション XQuery 評価 341 を参照してください。(xq アイコンは XQuery 評価 341 のためで、xqu アイコンは XQuery Update 式 345 のために使用されます。)
- JSON ドキュメントのクエリの説明に関しては JSON のためのXQuery 式 397 とXSLT/XQuery を使用したJSON 変換 500 を参照してください。

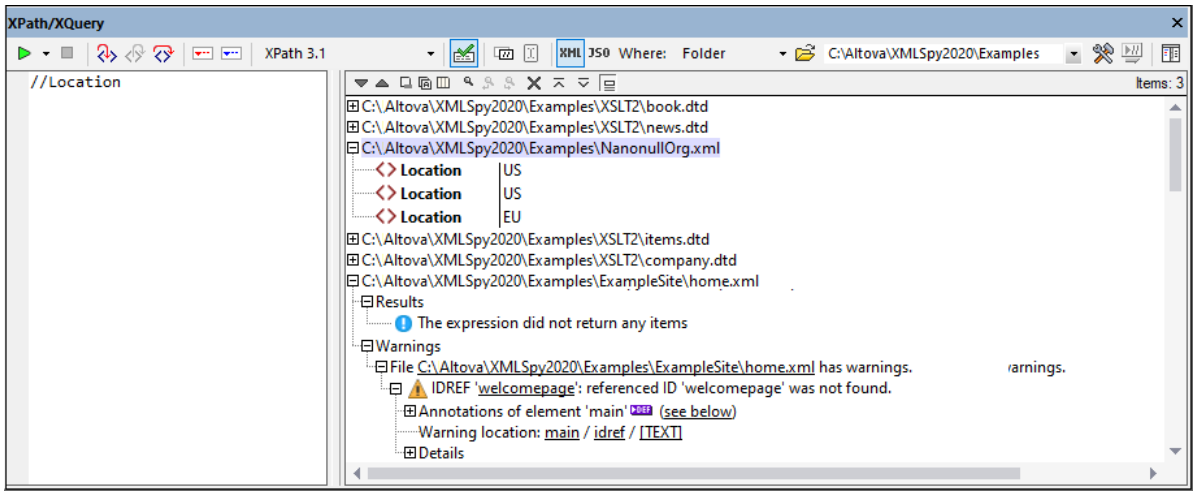
結果ペイン

結果ペインは下のスクリーンショット内の右側で表示されています。結果ペインには自身のツールバーが存在することに注意してください。



結果ペインには以下の機能が搭載されています:

- (*下のスクリーンショットで表示されているとおり*) 出力内でXML ファイルとXPath 式の場所を表示する場合 *出力内にヘッダーを表示する*アイコンをオンに切り替えます。
- 結果リストは2 つの列により構成されています: (i) ノード名、またはデータ型 (ii) ノードのコンテンツ。
- XPath 式が(要素または属性などの) ノードを返す場合、ノードのコンテンツ全体がノードの値として表示されるかを選択することができます。これを行うには、*完全な結果の表示*をオンに切り替えます。
- 式生成リテラルとは異なり結果が(テキストノードを含む)ノードを含む場合、結果ペイン内でそのノードをクリックするとメインウィンドウ内でXML ドキュメント内の対応するノードがハイライトされます。
- (*選択個所オプションで指定されている*) 複数のファイルに対して評価が行われる場合、各ファイルの結果はそのファイルのパスの下に個別にリストされます(*下のスクリーンショットを参照*)。1 評価モードがXML の場合、XML-適合ファイルは評価され、他の型はスキップされます。評価モードがJSON の場合、JSON-適合ファイルが評価され、他の型はスキップされます。



- 結果が複数のファイルを含む場合、結果リスト内でファイル名をクリックすると、XMLSpy 内でファイルが開かれ、このファイルがアクティブなファイルになります。
- 結果サブラインの両方の列をコピーまたは、値列のみをコピーすることができます。全ての列をコピーするには、サブラインを右クリックして、「**全ての列を含みコピーする**」をオンに切り替えます(または、コマンドを結果ペインのツールバー内のアイコンをオンとオフに切り替えコマンドを切り替えることができます。) コピーするサブラインを右クリックして(サブラインのための)「**サブラインのコピー**」または(全てのサブラインのための)「**全てをコピー**」を選択します。

結果ペインのツールバー

結果ペインのツールバーにはナビゲート、検索、およびコピー機能を提供するアイコンが含まれています。これらのアイコンは左から以下のテーブルで説明されています。対応するコマンドは結果リストアイテムのコンテキストメニューで使用することができます。

アイコン	動作

次、前	結果リストからそれぞれ次のおよび前のアイテムを選択します。
クリップボードに選択されたテキストラインをコピーする	クリップボードに選択された結果アイテムの値カラムをコピーします。全てのカラムをコピーするには、全てのカラムを含みコピーするコマンドに切り替えてください（*下を参照*）。
全てのメッセージをクリップボードにコピーする	空の値を含む値カラムのすべての結果アイテムをクリップボードにコピーします。各アイテムは異なるラインにコピーされます。
全てのカラムを含みコピーする	(i) 全てのカラム、または(ii) 値カラムのコピー間を切り替えます。カラムの区切りは単一のスペースです。
検索する	結果リスト内で特別文字を含む文字列を検索する検索ダイアログを開きます。
前を検索する	検索ダイアログ内に最後に入力された用語の前の発生を検索します。
次を検索する	検索ダイアログ内に最後に入力された用語の次の発生を検索します。
クリア	結果のリストをクリアする
複数のラインの結果を単一のラインに折りたたむ	結果アイテムの値カラムが複数ラインのテキストを含む場合、（新しいラインまたは文字を含む場合）、複数ラインと単一ラインの表示を切り替えることができます。
完全な結果を表示する	ノードの値としてノードのコンテンツ全体を表示します。
出力内にヘッダーを表示する	XML ファイルとXPath 式のロケーションを表示します。

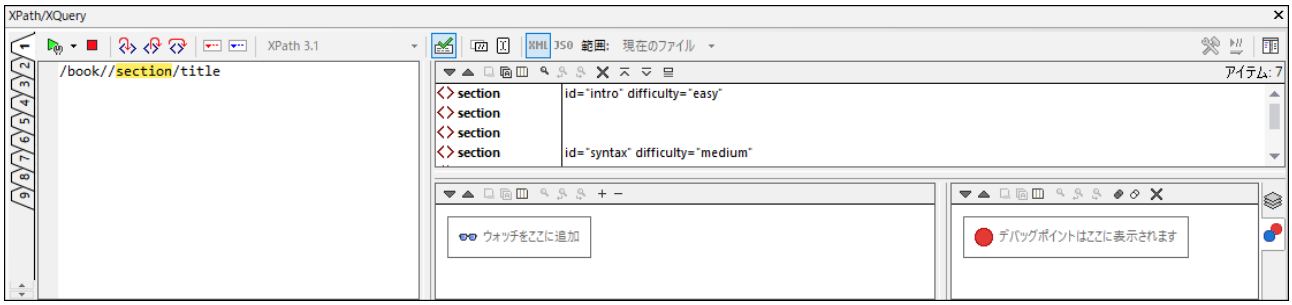
(C) 2015-2021 Altova GmbH

10.3 式のデバッグ

XPath/XQuery ウィンドウのデバッグモードによりアクティブなファイルのコンテキスト内でXPath/XQuery 式をデバッグすることが可能になります（*下のスクリーンショット*）。

デバッグモード内では結果ペインに2つのペインが追加されました（*下のスクリーンショットを参照*）:

- ペイン内で個別のタブを有するコールスタックとデバッグポイントペイン。
- ウォッチ式と変数が同じペインに表示されている変数とウォッチペイン。



式と結果ペインは水平的、または垂直的に隣接することができます。これらのレイアウトを切り替えるには、（ウィンドウのツールバーの右上の）「**水平的な/垂直的なレイアウト**」ボタンをクリックします。

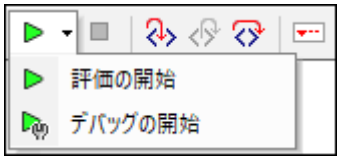
デバッガーモードには以下の機能が搭載されています:

- XPath 式がどのように評価されているか確認するためにXPath 評価プロセスにステップインすることを可能にします。**ステップイン (F11)** ツールバーボタンをこの目的にために使用します。各評価ステップでは、式の一部が現在評価されている場合は黄色でハイライトされますがそのステップの評価の結果は結果ペイン内で表示されます（*上のスクリーンショットを参照*）。例えば、上のスクリーンショットでは、`book` 要素のすべての`section` 子孫要素が選択されています。
- 評価を一時停止する箇所にブレークポイントを設定し、この箇所の結果を確認します。評価をブレークポイントでのみ一時停止してステップスルーすることができます。このために**デバッグの開始**（F5）ツールバーボタンを使用します。これは**ステップイン**（F11）を使用して全てのステップで一時停止するよりも素早く評価を行うことができます。を使用してください。
- トレースポイントを設定してトレースポイントとしてマークされたステップでの結果のレポートを確認することができます。評価は（ブレークポイント以外）一時停止せず、トレースポイントの結果は結果ペイン内のリストに表示されます。
- ウォッチ式は（評価のドキュメントデータまたはアスペクトなど）情報を確認するために使用することができます。これはブレークポイントに対して特に役に立ちます。
- 自身の値を含むスコープ内の変数は変数とウォッチ式ペイン内に表示されます。
- 評価ステップのプロセッサーの呼び出しはコールスタックとデバッグペインのコールスタックタブ内で表示されています。
- ブレークポイントとトレースポイントが設定されている場合、これらはコールスタックとデバッグペインのデバッグポイントタブに表示されます。

これらの機能の詳細に関しては、下の説明を参照してください。

デバッグモードのセットアップ

デバッグモードを切り替えるには、ツールバーの左上にある「**評価/デバッグの開始**（F5）」コマンドのドロップダウンメニュー内の「**デバッグの開始**」をクリックします（*下のスクリーンショットを参照*）。デバッグモード内では、ウォッチ式、コールスタック、およびデバッグポイントペインが表示されます。現在の式のデバッグを開始するには、「**デバッグの開始**」または「F5」をクリックします。



現在のファイルでのみデバッグモードは作動することに注意してください。複数のファイルとは使用することができません。この結果、選択個所が*現在のファイル*に設定されていると、このモードに切り替えられます。

デバッグモードをセットアップするためのボタン

	評価/デバッグの開始 (F5)	デバッガーを開始します。
	水平的な/垂直的なレイアウト	水平的および垂直的レイアウト間の切り替えを行います。
	ビルダーに切り替える	式の作成を助けるコンテキストを考慮した入力ヘルパーを提供する式ビルダーに切り替えます。
	現在の選択範囲の XPath をコピーする	XML ドキュメント内のノードのロケーターパスを式ペイン内のカーソルの最後の場所にコピーします。
XML JSO	**XML/JSON 評価モード**	ペアのハイライトされているアイコンはアクティブなオプションです。評価モードはファイル型に従い自動的に検知されます。他のアイコンは無効化されます。
	現在の選択範囲をコンテキストとして設定する	ルートノードと現在の選択範囲の式コンテキストを切り替えます。
	入力中に評価	式の入力中に式の評価がオンに切り替えられます。
	XML の評価	オンに切り替えられるとターゲット XML ドキュメントが検証されます。

デバッガーの実行

XPath/XQuery 式のデバッグのためのステップは通常以下で説明される通りです:

1. 式を実行する XML/JSON ファイルをアクティブなファイルにします。
2. XPath/XQuery 式を式ペインに入力します。
3. ブレークポイントまたはトレースポイントを設定します。ブレークポイントは評価が一時停止されるポイントです。トレースポイントは評価が記録されるポイントです。ですからトレースポイントは評価結果をトレースすることが可能なパスを提供します。
4. 「**デバッガーの開始**」,をクリックすると、式内でブレークポイントがマークされていない限り、評価が終わりまで1つのステップで実行さ「**デバッガーの開始**」を繰り返しクリックして、各ブレークポイントを前進し評価の終わりまで到達します。
5. ステップイン/アウト/オーバー機能を使用してステップバイステップに評価を進めることができます。

デバッグのためのボタン

	デバッガーの開始 (F5)	デバッガーを開始します。ブレークポイントで停止する以外、評価は直接最後まで行われます。
	デバッガーの停止 (Shift+F5)	評価を終了しデバッガーを停止します。

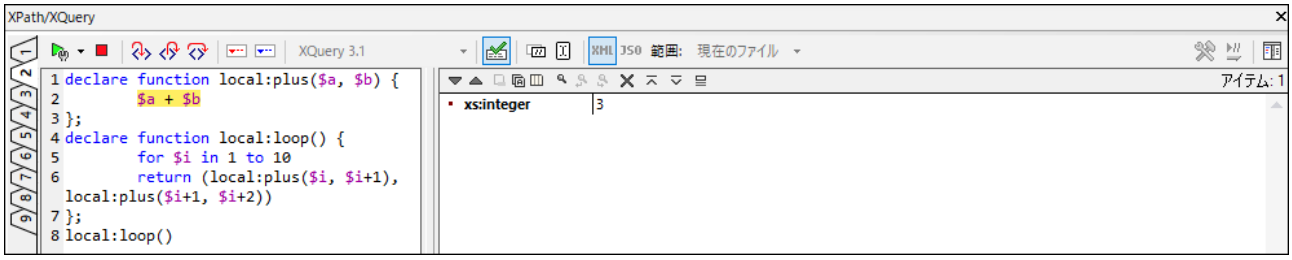
	ステップイン（F11）	評価をステップごとに進めます。
	ステップアウト（Shift+F11）	現在の評価ステップをステップアウトし、親ステップに移動します。
	ステップオーバー（Ctrl+F11）	子孫ステップにステップオーバーします。
	ブレークポイントの挿入/削除（F9）	カーソルが置かれている場所の式ステップでブレークポイントを挿入/削除します。
	トレースポイントの挿入/削除 (Shift+F9)	カーソルが置かれている場所の式ステップでトレースポイントを挿入/削除します。

評価ステップのステップイン、アウト、オーバー
ステップイン 機能は評価をステップバイステップで行うことができます。このコマンドは評価の次のステップに移動します。現在のステップは式内でハイライトされて表示されています（*下のスクリーンショットを参照*）。*ステップアウト* 機能は現在のステップを高いレベルに移動しますが、*ステップオーバー* 機能は同じレベルより1つ低いレベルに移動します。下のXQuery 3.1 式を式ペインに張り付け *ステップ* ボタンをクリックしてステップ機能をテストすることができます。

ステップイン、ステップアウト、およびステップオーバー機能を試す XQuery 3.1 式

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

下のスクリーンショットは $i=1 である最初のパススルーループ中に追加ステップに到達した際に処理が一時停止された評価を表示しています。この追加ステップでは、結果は（1+2 の結果である）3 を表示します。



ブレークポイント

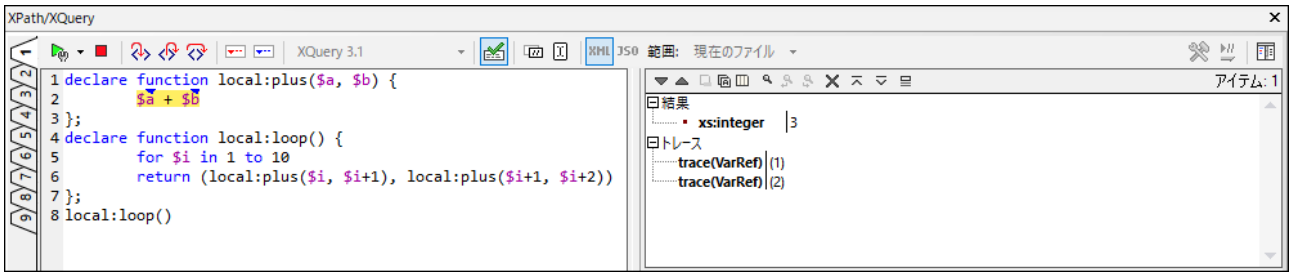
ブレークポイントは「**デバッガーの開始**」を使用して開始された後デバッガーを停止するポイントです。デバッガーがブレークポイントで停止すると、結果をチェックして「**ステップイン**」機能を使用し評価の次のステップの結果を表示することができます。ブレークポイントを設定するには、ブレークポイントを設定する箇所でカーソルを式内にポイントし、「**ブレークポイントの挿入/削除（F9）**」ツールバーボタンをクリックします。ブレークポイントは赤い点線のオーバーラインで表示されます。ブレークポイントを削除するには、選択し「**ブレークポイントの挿入/削除（F9）**」をクリックします。

デバッグポイント 393 を参照してください。

トレースポイント

トレースポイントは結果が記録されるポイントです。これらの結果は*結果*タブの*トレース*ツリー内で表示されます（*下のスクリーンショットを参照*）。これにより式の特定の箇所のすべての評価結果を確認することができます。例えば、下のスクリーンショット内では、トレースポイントは `$a`, `$b`, と`local:plus($i, $i+1)`; に設定されており、ループの最初の反復中のこれらのトレースポイントでの結果は以下の*トレース*ツリーに表示されます: `$a=1`, `$b=2`, と`local:plus($i, $i+1)=3`.

トレースポイントを設定するには、トレースポイントを設定する箇所でカーソルを式内にポイントし、ツールバーボタン「**トレースポイントの挿入/削除（Shift+F9）**」をクリックします。トレースポイントは赤い点線のオーバーラインで表示されます。（*下のスクリーンショットを参照*）. トレースポイントを削除するには、選択し「**トレースポイントの挿入/削除（F9）**」をクリックします。

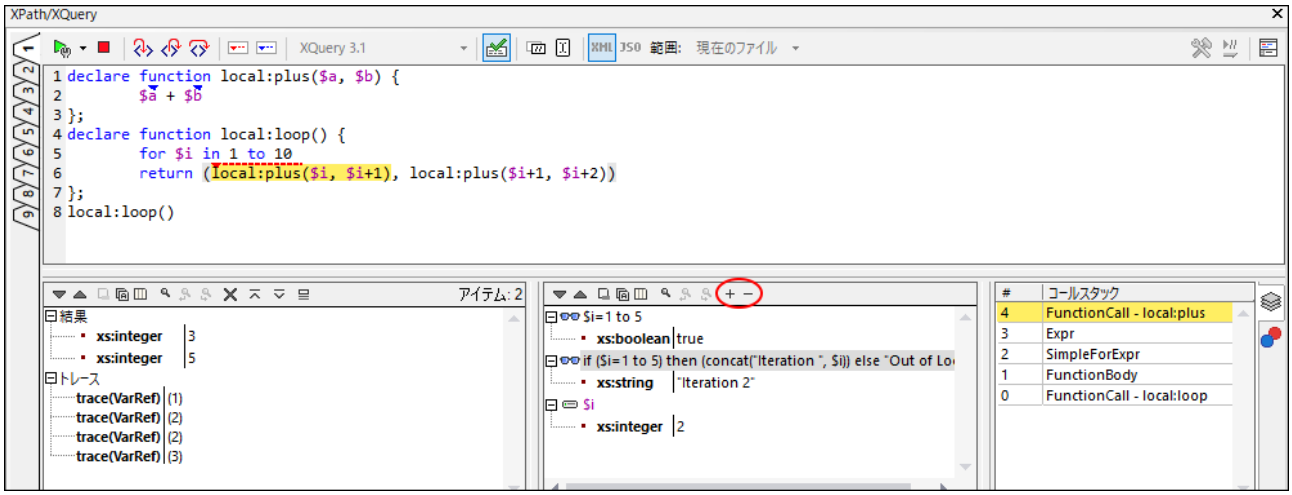


メモ ブレークポンとトレースポイントがセットで式の一部である場合、オーバーラインは赤と青の点線で示されています。

デバッグポイント 393 を参照してください。

変数、ウォッチ式、およびコールスタック

変数とウォッチ式は変数とウォッチ式ペインに表示されます（*下のスクリーンショットの下の中央のペイン*）。



変数

式内で宣言されており、現在の評価ステップ内のスコープにある変数は対応する現在の値と共に表示されます。例えば、上のスクリーンショットでは、処理は`local:plus` 関数への呼び出しに達成しました。`$i` 変数は`local:loop` 関数のスコープ内にあり、2 に増やされました。ですから、`$i` は現在の値と共に表示されています。 `$a` と`$b` 変数は`local:loop` 内にはなく、このためスコープ内にはありません。この結果これらは表示されません。

ウォッチ式
評価の開始前、または、評価の一時中止中にウォッチ式を入力することができます。次の目的のために使用することができます：

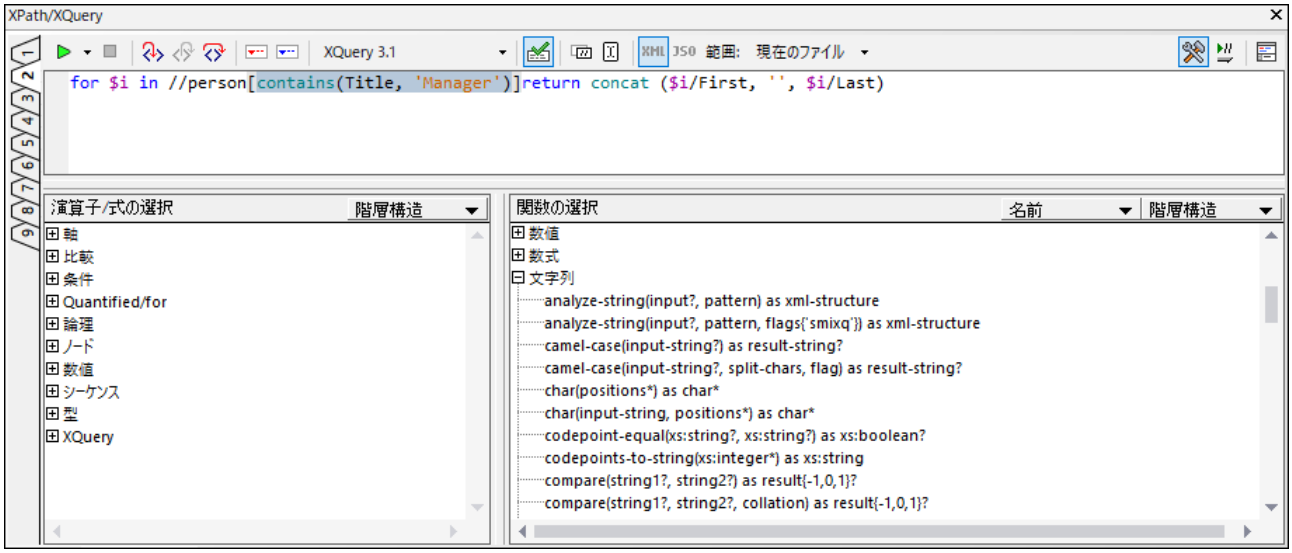
- 特定の条件をテストするため。例えば、上のスクリーンショットでは、ウォッチ式 `$i=1 to 5` は `$i` 変数が指定された範囲に値を持つかをテストするために使用されています。結果 `true` は条件が満たされたことを示しています。
- 特定のコンテキスト内のデータを検索するため。例えば、`Company` 要素のコンテキストでウォッチ式 `@id` を入力して、会社の顧客コードを XML ターゲットドキュメント内でルックアップすることができます。
- 追加データを生成するため。追加データを生成するため。例えば、上のスクリーンショットでは、ウォッチ式 `if ($i=1 to 5) then (concat("Iteration ", $i)) else "Out of Loop"` は現在の評価ループを反復する適切な文字列を生成することができます。

ウォッチ式を入力するには、ペインのツールバー内で*(上のスクリーンショットで赤い枠で囲まれてる)*「**ウォッチエントリの追加**」をクリックし、式を入力し、完了すると **Enter** をクリックします。ウォッチ式を削除するには、ツールバー内で選択し「**選択されたウォッチエントリを削除する**」をクリックします。デバッグ中に式が（例えば、変数の一つがスコープ外にあるなど）何らかの理由で正確に評価されない場合ウォッチ式は赤い色で表示されます。

コールスタック
コールスタックとデバッグポイントペインのコールスタックタブはその時点までのデバッグ内のプロセッサーの呼び出しを表示しています。*（上のスクリーンショット内の右下のペイン）*。現在のプロセッサーの呼び出しは黄色でハイライトされています。現在の評価ステップに直接導いた呼び出しが表示されます。例えば、上のスクリーンショットでは、現在の評価ステップは `local:plus` への関数呼び出しです。これは `local:loop` の2番目の反復ですが、最初の反復のプロセッサー呼び出しは表示されません。これはこれらの呼び出しは現在の関数呼び出しの並列レベルにあり、呼び出しを導いたわけではありません。

デバッグポイント

コールスタックとデバッグポイントペインのデバッグポイントタブは式に設定することのできるブレークポイント（赤丸）とトレースポイント（青丸）を表示します*（下のスクリーンショット内の右下のペイン）*。各デバッグポイントはラインと文字番号と共にリストされています。例えば、`FunctionCall@7:13` はライン7、文字13にデバッグポイントが存在することを示しています。



以下の機能に注意してください：

- ブレークポイントに関しては、*ブレーク条件の入力をダブルクリックして*、「**ブレークの条件**」を入力し、「**Enter**」を押して入力します。ブレークポイントは条件が `true` を評価すると有効化されます。例えば、上のスクリーンショットでは、ブレークの条件 `$i=3` は `$i` の値 3 がの場合のみこの関数呼び出し上でブレークポイントを有効化します。スクリーンショットは子のブレークポイントで一時停止された評価を表示しています。

- 全てのデバッグポイントの有効化/無効化を対応するツールボタンをクリックすることで行うことができます: **全てのデバッグポイントの有効化** と **全てのデバッグポイントの無効化** *(ボタンは上のスクリーンショット内でボタンは緑色の枠で囲まれています)*。デバッグポイントが無効化されると、有効化されるまで全て無効化されます。
- 対応するコンテキストメニュー内で個別のブレークポイントを有効化/無効化することができます。

ペイン内のツールバーコマンド

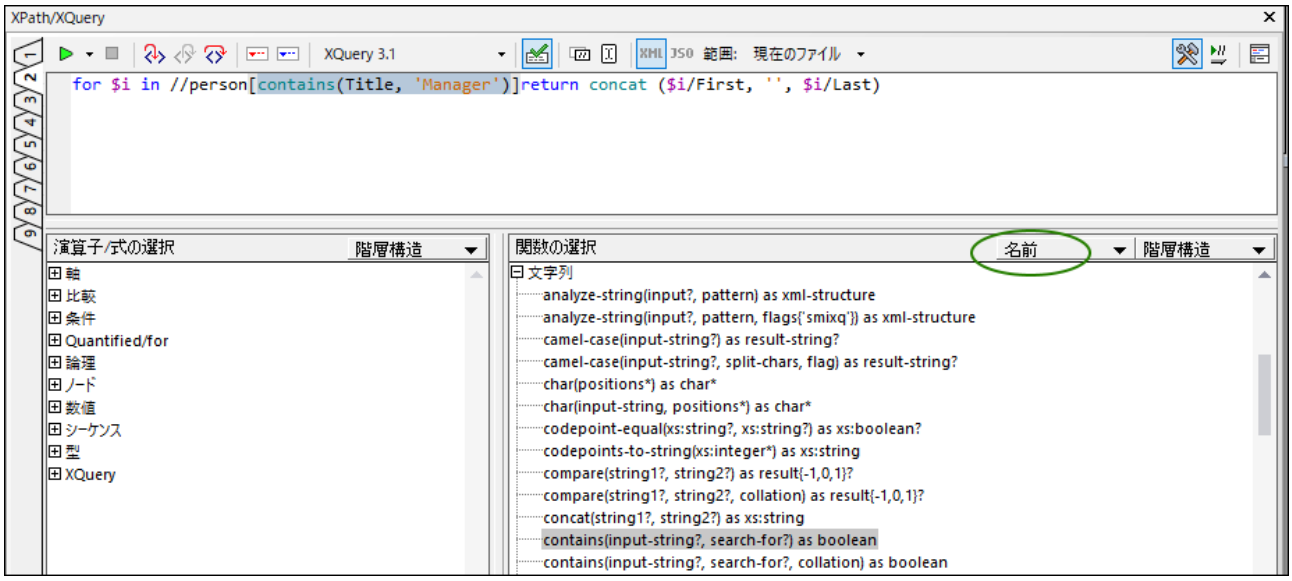
デバッグモード内のXPath/XQuery ウィンドウにはナビゲート、検索、およびコピー機能を含むボタンが搭載されています(*上のスクリーンショットを参照*)。これらのアイコンは左から以下のテーブルで説明されています。対応するコマンドは結果リストアイテムのコンテキストメニューで使用することができます。

アイコン	動作
次、前	結果リストからそれぞれ次のおよび前のアイテムを選択します。
クリップボードに選択されたテキストラインをコピーする	クリップボードに選択された結果アイテムの値カラムをコピーします。全てのカラムをコピーするには、全てのカラムを含みコピーするコマンドに切り替えてください(*下を参照*)。
全てのメッセージをクリップボードにコピーする	空の値を含む値カラムのすべての結果アイテムをクリップボードにコピーします。各アイテムは異なるラインにコピーされます。
全てのカラムを含みコピーする	(i) 全てのカラム、または(ii) 値カラムのコピー間を切り替えます。カラムの区切りは単一のスペースです。
検索する	結果リスト内で特別文字を含む文字列を検索する検索ダイアログを開きます。
前を検索する	検索ダイアログ内に最後に入力された用語の前の発生を検索します。
次を検索する	検索ダイアログ内に最後に入力された用語の次の発生を検索します。
クリア	結果のリストをクリアする
複数のラインの結果を単一のラインに折りたたむ	結果アイテムの値カラムが複数ラインのテキストを含む場合、(新しいラインまたは文字を含む場合)、複数ラインと単一ラインの表示を切り替えることができます。
完全な結果を表示する	ノードの値としてノードのコンテンツ全体を表示します。
出力内にヘッダーを表示する	XML ファイルとXPath 式のロケーションを表示します。

(C) 2015-2021 Altova GmbH

10.4　式ビルダー

式ビルダー（または、ビルダーモード）はメインツールバー の「**ビルダーモード**」アイコンをクリックしてオンとオフに切り替えます（*下のスクリーンショットを参照*）。式ビルダーは両方のモードでオンに切り替えることができます（評価モード とデバッグモード 382）。2つの入力ヘルパーペインが存在します: (i) 演算子と式、および(ii) 関数（*下のスクリーンショットを参照*）。両方のペイン内のアイテムは階層的にまたはフラットリストとして表示されます。各ペインの右上にあるドロップダウンリスト内からオプションを選択します（*下のスクリーンショットを参照*）。スクリーンショット内では、アイテムは階層的なグループにより表示されています。



式ビルダーの機能:

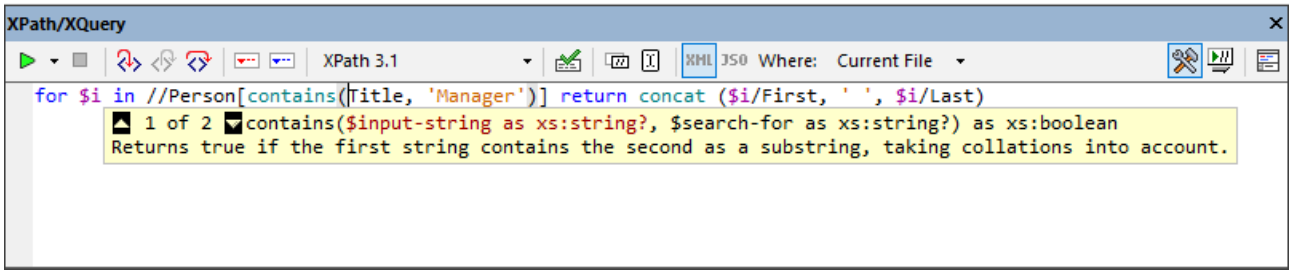
- 入力ヘルパーペイン内のアイテムのテキスト詳細を確認するためには、マウスをアイテムにポイントしてください。
- 各関数は署名（すなわち、引数、引数のデータ型、および関数の出力のデータ型）と共にリストされています。
- 1つ以上の署名が単一の関数名のために存在する場合、各署名は個別の関数としてリストされています。（これらのバリアントはその関数名のオーバーロード として知られています。）上のスクリーンショットでは、例えば、`contains` 関数は2度表示されています: 2つの署名のためにそれぞれ表示されています。
- 引数は名前により、またはデータ型によりリストされています。関数ペインのタイトルバー内ののドロップダウンリストからオプションを選択します（*上のスクリーンショット内では緑色の枠で囲まれています。*）
- ペイン（演算子、式、または関数）内のアイテムをダブルクリックして、式内のカーソルの位置にそのアイテムを挿入します。式はプレースホルダ（`#` シンボルにより示される引数と共に挿入されます。
- (i) テキストが式の編集フィールドで選択されており（ii) プレースホルダを含む演算子、式または関数はダブルクリックして挿入することができます。選択済みのテキストはプレースホルダの代わりに挿入されます。これは（パス式などの）長いテキストを演算子、式、または関数に挿入する素早い方法です。
- ノードを選択しウィンドウのメインツールバーのボタン「**現在の選択範囲の XPath をコピーする**」をクリックしてアクティブドキュメント内のノードにパスを挿入することができます。

式ビルダー内で使用されるツールバーボタン

	水平的な/垂直的なレイアウト	水平的および垂直的なレイアウト間を切り替える

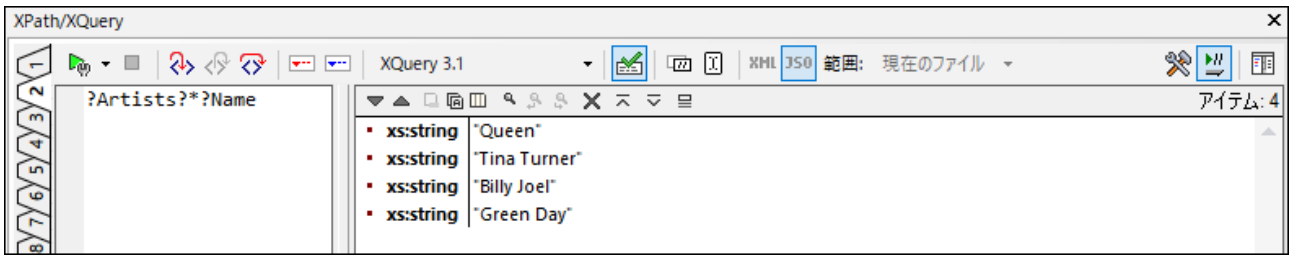
	ビルダーに切り替える	式を作成するためのコンテキストを考慮した入力ヘルパーを提供する式ビルダーモードに切り替える。
	現在の選択の XPath をコピーする	XML ドキュメント内のノードのロケーターパスを式ペイン内の最後のカーソルの位置にコピーします。
	コンテキストとして現在の選択範囲をセットする	ルートノードと現在の選択範囲間の式コンテキストを切り替えることができます。

式内に関数を入力した後、式ペイン内の関数名にカーソルをポイントすると関数の署名と関数のテキストの詳細が表示されます。関数のために1つ以上の署名が存在する場合、これらは表示の下のオーバーロード要素を共に示されます。関数の括弧内にカーソルをポイントし「Ctrl+Shift+Spacebar」を押すと、その関数名の多種のオーバーロードの署名を確認することができます（*下のスクリーンショットを参照*）。



10.5 JSONX のための Query 式

JSON ドキュメントはXPath/XQuery 出力ウィンドウ[108] 内にXPath/XQuery 3.1 クエリ式を入力するとクエリすることができます（*下のスクリーンショット参照*）。



JSON ドキュメントを評価するには以下を行います:

1. **XPath 3.1** アイコンまたは**XQuery 3.1** アイコンを選択します。
2. ウィンドウがJSON 評価モードであることを確認します。
3. XPath 3.1 またはXQuery 3.1 式を入力します。
4. （ツールバーの左の）「**評価の開始**」をクリックします。

JSON のためのXQuery 3.1 式

JSON データ構造は通常オブジェクトと配列を使用するため、 XQuery 3.1 lookup operator `?` が（基本的にXQuery パースペクティブからマップする） JSON オブジェクトとJSON 配列内のノードを検索するために使用されます。ノードをロケートするこの方法はXML ドキュメント内でノードをロケートするためにパス式が作成される方法とは異なります。これらの中では、スラッシュ演算子 `/` がパス式内でステップを接続するために使用されています（例: `items/*`）。JSON のための式 XQuery 内では、スラッシュ演算子はノードをロケートするために使用されません。

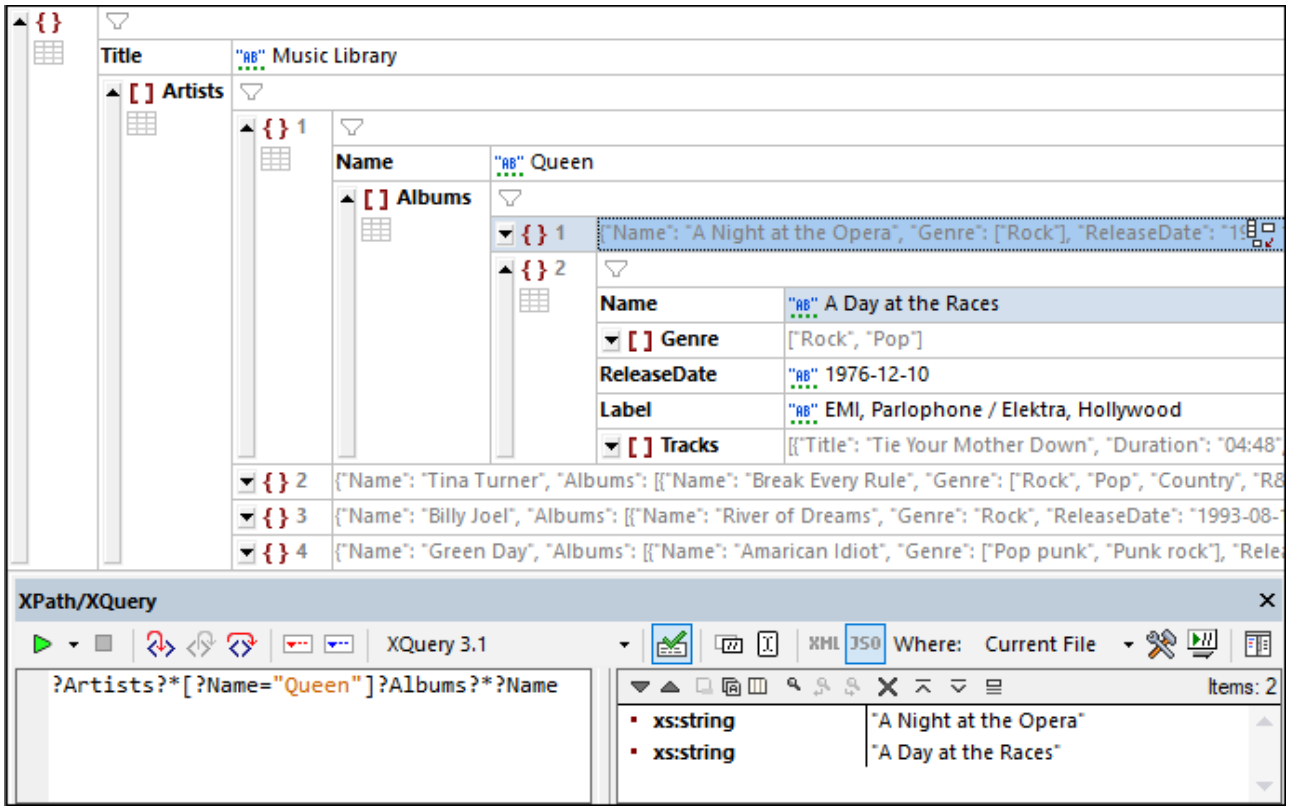
JSON のためのXQuery 式のサンプル

`?items?*`
意味: 子ノード `items` をルックアップし、自信の子ノード全てをルックアップします。`items` はコンテキストノードの子ノードであることが期待されることに注意してください。

`?Artists?1?Albums?2?Name`
意味: 子ノード `Artists` をルックアップし、最初の子ノードをルックアップします。そのノードの内部で子ノード `Albums` をルックアップし、2 番目の子ノードをルックアップします。2番目の`Name` ノードが返されます。

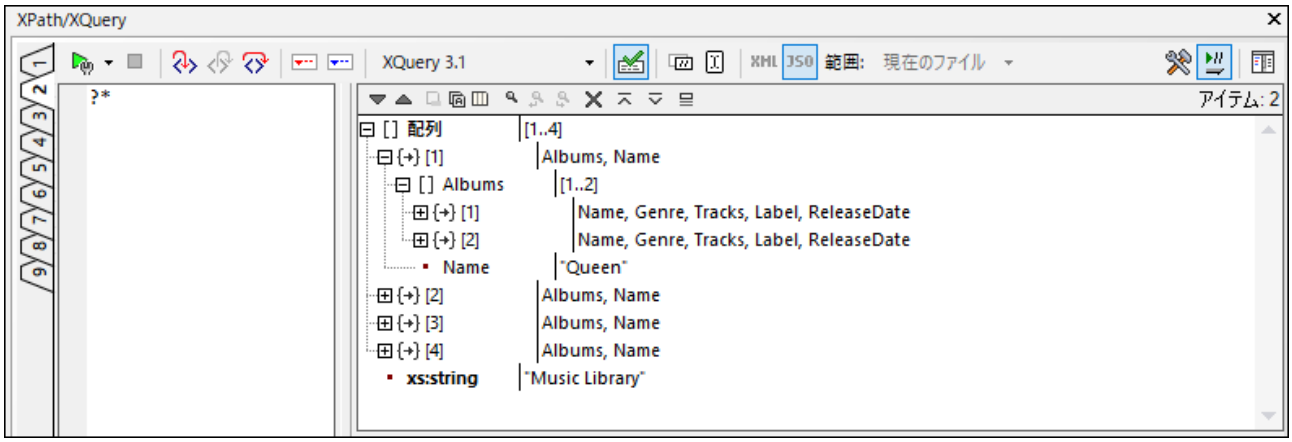
`?Tracks?*[contains(?Writer, 'Brian')]`
意味: 子ノード `Tracks` をルックアップし、自信のすべての子をルックアップします。子をルックアップ中に、各 `Writer` 子ノードの子をルックアップし、文字列 `'Brian'` を含むものを選択します。この式内には3 つのルックアップ演算子が存在します。ノードセットがルックアップされる個所でそれぞれが新規ステップ内で使用されています。

`?Artists?*[?Name="Queen"]?Albums?*?Name`
意味: ルートオブジェクト内でlookup 子ノード `Artists` をルックアップし、`"Queen"` の値を持つ`Name` ノードを持つすべての子をルックアップします。コレアのノード内で、全ての子 Albums ノードをルックアップし、その子もルックアップします。これらの子内で、対応する`Name` ノードをルックアップ（して返）します。下のスクリーンショットでは、この式はXPath/XQuery ウィンドウ[381] 内でターゲット JSON ドキュメントのJSON グリッドビュー[475] 表示と共に表示されています。



JSON 評価のための結果ペイン

結果は選択された JSON コンポーネントを*結果* ペインの左側に太字で表示し、コンポーネントの値をペインの右側に表示します。下のスクリーンショットでは、結果は太字で表示されます。配列はメンバーを表示するために展開されています。



完全な結果の表示 アイコンがツールバー内でオンまたはオフに切り替えられているかに従い、マップと配列は短い、または詳細フォーマットで表示されています。ペインの左側のマップと配列はそれぞれのプラスアイコンをクリックしてて展開することができます。

10.6 注意点

XPath 1.0 条件式

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

XPath 2.0 と3.1 条件式

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

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

XPath 2.0 と3.1 内のデータ型

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

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

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

文字とエンティティ参照の文字列長

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

XPath 2.0 および 3.1 関数サポート

付録 1488 を参照ください。

(C) 2015-2021 Altova GmbH

11 Authentic

(以下のスクリーンショットにある) Authentic View はお使いの XML ドキュメントをグラフィカルな形式で表示したものです。XML ドキュメントのマークアップの表示を隠し、入力フィールド、コンボボックス、ラジオボタンといったフォーマットやデータ入力機能を適切な形で表示します。Authentic View を使ってユーザーが入力したデータは XML ファイルにも反映されます。

Authentic View でデータの閲覧と入力を行うためには、Altova StyleVision により作成された **StyleVision Power Stylesheet (SPS)** に XML ドキュメントを関連付ける必要があります。SPS (.sps ファイル) は、実質的には XSLT スタイルシートで、データ入力を行うことができる XML ファイルの出力フォーマットを指定することができます。Authentic View ユーザーは、XML ファイルや DB へのデータ入力をグラフィカルに行うことができます。SPS はスキーマをベースにしており、スキーマに特化したファイルです。Authentic View ビューで SPS を使った XML ファイルの編集を行うには、XML ファイルがベースにしているスキーマファイルを使用しなければなりません。

Authentic View を使う

- XML ファイルが開かれている場合、メインウィンドウの下部にある **Authentic** ボタンをクリックすることで Authentic View に切り替えることができます。SPS が XML ファイルにまだ割り当てられていない場合、割り当てを行うよう促されます。使用できる SPS は、XML ファイルがベースにしているスキーマをベースにしたもので無ければなりません。
- **「ファイル | 新規作成」** コマンドを選択し、「StyleVision スタイルシートを選択」ボタンを選択すると、新規 XML ファイルが作成され、Authentic View で表示されます。この新しいファイルは、選択した SPS ファイルに割り当てられたテンプレートファイルになります。このファイルは、既に表示されている様に様々な数の開始データを持つことができます。この開始データは (テンプレート) XML ファイルに含まれ、SPS ファイルと関連付けることもできます。XML ファイルの Authentic View が表示されたら、データの入力とファイルの保存を行うことができます。

- 「**Authentic | 新規作成**」 コマンドを使用してSPS を開くことができます。テンプレート XML ファイルがSPS に割り当てられている場合、テンプレート XML ファイル内のデータは、Authentic View.により作成されたXMLドキュメントテンプレートのデータを開始するために使用されます。

このセクションでは..

このセクションはAuthentic ビューのチュートリアルを含んでおり、Authentic View の使い方を説明した後、Authentic ビューでの編集や個々の編集機能の詳細について説明します。

Authentic View について更に詳しく

Authentic View に関する更に詳しい情報については、(i) Authentic View 編集ウィンドウについて解説された、このドキュメントの **Authentic**[401] | Authentic ビューインターフェイス[415]を参照するか、(ii) このドキュメントのユーザーリファレンスにある Authentic メニュー[901]セクションを参照ください。

11.1 Authentic View チュートリアル

Authentic View 内の視覚的な W YSIW YG インターフェイス内で Microsoft W ord などのワードプロセッサーアプリケーション内で作業するように、XML ドキュメントを編集することができます（*下のスクリーンショット*）。XML ドキュメントの Authentic View を管理するスタイルシート内で既に定義されているため、ドキュメントの書式設定を心配する必要はありません（StyleVision Power Stylesheet はこのチュートリアル内では SPS と省略されています）。スタイルシートは Altova の StyleVision 製品のスタイルシートデザイナーを使用して作成されます。

Authentic View 内で XML ドキュメントを編集するには2 つのユーザーアクションが含まれます: (i) ドキュメントの構造の編集（例えば、段落とヘッドラインなどのドキュメントの一部を追加、または、削除するなど）、および (ii) データの挿入（ドキュメントの一部のコンテンツ）.

このチュートリアルでは次のステップについて説明されています:

- Authentic View 内で XML ドキュメントを開く。Authentic View 編集の主要な必要条件は、XML ドキュメントが SPS ファイルに関連付けられていることです。
- Authentic View インターフェイスの主要な編集メカニズムに関する概要を確認してください。
- ノードを挿入または削除することによりドキュメントの構造を編集します。
- XML ドキュメント内にデータ挿入します。
- (i) 属性入力ヘルパーを介した属性の値と (ii) エンティティの値を入力します。
- ドキュメントを印刷します。

このチュートリアルは使用開始の準備をするためであり、簡単にまとめられています。Authentic View インターフェイス 415 セクション内で追加の参照マテリアルと機能の説明を確認することができます。

チュートリアルの必要条件

チュートリアルで必要な全てのファイルはAltova アプリケーションの `Examples` フォルダー内にあります。これらのファイルは、以下のとおりです:

- NanonullOrg.xml (XML ドキュメント)
- NanonullOrg.sps (XML ドキュメントがリンクされている StyleVision Power Stylesheet)
- NanonullOrg.xsd (XML ドキュメントと StyleVision Power Stylesheet がベースにされ、リンクされている XML スキーマ)
- nanonull.gif and Altova_right_300.gif (チュートリアルで使用されるイメージ)

メモ このチュートリアルのある時点で、(ドキュメントの Authentic View とは異なり) XML ドキュメントの XML テキストを確認するように促されます。(Authentic Desktop と Authentic Browser 同様) Altova 製品エディションにテキストビューが搭載されていない場合、Wordpad または Notepad のようなテキスト エディターを使用して XML ドキュメントのテキストを確認してください。

注意点: 問題が発生した際に元のファイルにアクセスできるように、`NanonullOrg.xml` をチュートリアルのためにコピーすることが奨励されます。

11.1.1 AUTH-VIEW 内で XML ドキュメントを開く

Authentic View 内で、既存の XML ドキュメントを編集、または、新規の XML ドキュメントを作成して編集することができます。(このセクションで説明されているとおり) このチュートリアルでは、Authentic View 内で既存の XML ドキュメントを開き、(次のセクションで) 編集の方法を学びます。更に、Authentic View 内での編集のために新規 XML ドキュメント作成の方法について説明されています。

既存の XML ドキュメントを開く

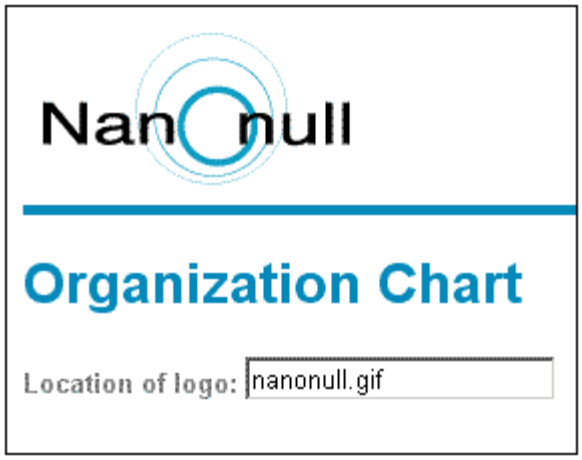
`NanonullOrg.xml` というファイルを開きます。Altova アプリケーションの `Examples` フォルダー内にあり、以下の2つの方法により `NanonullOrg.xml` を開くことができます:

- Altova 製品内の「**ファイル | 開く**」をクリックし、表示されるダイアログ内から NanonullOrg.xml を選択し、「**開く**」をクリックします。
- ウィンドウ Explorer を開き、ファイルを検索し、右クリックして、ファイルを開くためのアプリケーションとして Altova 製品を選択します。

Authentic View 内でファイル `NanonullOrg.xml` を直接開くことができます(*下のスクリーンショット*)。理由:

1. StyleVision Power Stylesheet (SPS) が既にファイルに割り当てられており、
2. SPS ファイルの割り当てが既に割り当てられている場合、表示タブのオプションダイアログ(「**ツール | オプション**」)で、Authentic View 内で XML ファイルを開くオプションを選択します(それ以外の場合、ファイルはテキストビュー内で開かれます)。

(C) 2015-2021 Altova GmbH



留意点: SPS はAuthentic View 内でXML ドキュメントがどのように表示されるかを表示し管理します。SPS が存在しない場合、ドキュメントのAuthentic View は存在しません。

SPS をベースにした新規のXML ドキュメント

SPS をベースにした新規のXML ドキュメントを作成することができます。これを2 つの方法で行うことができます: **「ファイル | 新規作成」**メニューコマンドと**「Authentic | 新規のドキュメント」**メニューコマンドを使用する。SPS が両方の方法で選択されます。

「ファイル | 新規作成」を使用する方法

1. **「ファイル | 新規作成」**を選択します。
2. 新規のドキュメントの作成ダイアログ内で、希望するSPS を参照します。

テンプレート XML ファイルがSPS に割り当てられている場合、Authentic View 内で作成されたドキュメントテンプレートの開始データとしてテンプレート XML ファイル内のデータが使用されます。

「Authentic | 新規のドキュメント」を使用する方法

1. **「Authentic | 新規のドキュメント」**を選択します。
2. 「新規のドキュメントの作成」ダイアログ内で希望するSPS を参照します。

テンプレート XML ファイルがSPS に割り当てられている場合、テンプレート XML ファイル内のデータは、Authentic View. 内で作成されるXML ドキュメントテンプレートの開始データとして使用されます。

11.1.2 Authentic View インターフェイス

Authentic View 編集インターフェイスはドキュメントデータを入力し編集するメインウィンドウと2 つの入力ヘルパーにより構成されています。ドキュメントの編集は簡単です。ドキュメントのマークアップを確認する場合、マークアップタグをオンに切り替え、ドキュメントのコンテンツの入力を開始します。ドキュメントの構造を変更するには、コンテキストメニュー、または、要素入力ヘルパーを使用します。

XML ノードタグ(ドキュメントマークアップ)の表示

XML ドキュメントは、基本的には、ノードの階層構造です。例:

<ドキュメントRoot>

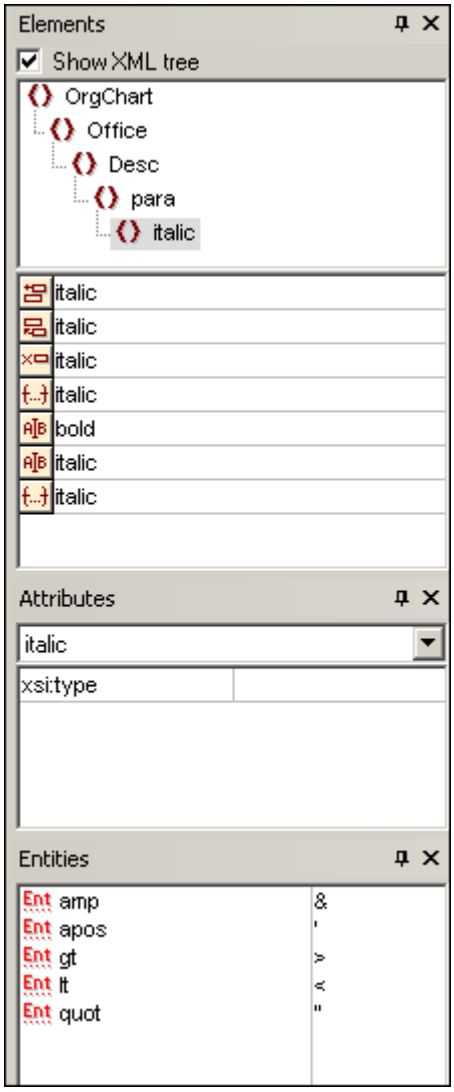
```
  <Person id="ABC001">
        <Name>Alpha Beta</Name>
        <Address>Some Address</Address>
        <Tel>1234567</Tel>
    </Person>
</ドキュメントRoot >
```

デフォルトでは、ノードタグ Authentic View 内では表示されていません。メニューアイテム「Authentic | 大きなマークアップを表示する」（または、 ツールバーアイコン）を選択してノードタグをオンに切り替えることができます。大きなマークアップタグにはそれぞれのノードの名前が含まれています。更に、（タグ内にノード名の無い）小さなマークアップと複合型マークアップを選択することができます（スタイルシートのデザイナーにより大きい、小さい、または、マークアップタグの無い複合型マークアップが定義されます。ドキュメントのためのデフォルトの複合型マークアップにはマークアップが存在しません）。

Altova 製品 またはテキストエディターのテキストビュー内で XML ドキュメントのテキストを確認することができます。

入力ヘルパー

インターフェイス内には、デフォルトでアプリケーションウィンドウの右端にある3つの入力ヘルパーが存在します（*下のスクリーンショット*）。これらは、要素、属性、および、エンティティ入力ヘルパーです。



要素入力ヘルパー: 要素入力ヘルパーは、カーソルの現在の場所、または、メインウィンドウ内の選択された場所に挿入、または、から削除することができる要素を表示します。入力ヘルパーは状況依存であることに注意してください。コンテキストはカーソルの位置、または、選択範囲により異なります。入力ヘルパーのコンテンツは以下の方法で変更することができます: 要素入力ヘルパーのXML ツリー内で他のノードが選択された場合、そのノードに関連する要素が入力ヘルパー内で表示されます。入力ヘルパーの上のXML ツリーの表示チェックボックスをチェックして、XML ツリーを表示するために要素入力ヘルパーを展開することができます（*上のスクリーンショットを参照してください*）。XML ツリーは、トップレベル要素ノードからマインウィンドウ内で選択されたノードまでのノードの階層構造を表示します。

属性入力ヘルパー: メインウィンドウ内で選択されている要素の属性、および、属性の値を表示します。属性の値は、属性入力ヘルパー内に入力し編集することができます。属性入力ヘルパーのコンボボックス内の選択のためにトップレベルの要素選択された要素までの要素ノードを使用することができます。

コンボボックスのドロップダウンリストから要素を選択すると、要素の属性が入力ヘルパー内に表示されます。入力ヘルパー内で要素の属性を編集することができます。

エンティティの入力ヘルパー: エンティティの入力ヘルパーは状況依存ではなく、ドキュメントのために選択された全てのエンティティを表示します。エンティティをダブルクリックすると、カーソルの位置にエンティティを入力することができます。ドキュメントへのエンティティの追加方法はセクション Authentic View インターフェイス 415 で説明されています。

コンテキストメニュー

Authentic View ドキュメント内を右クリックすると、その場所（ノード）に関連したコンテキストメニューがポップアップされます。コンテキストメニューは以下を行うことのできるコマンドを提供します:

- 選択されたノードの前、または、あとにノードを挿入します。サブメニューはそれぞれの挿入場所で使用を許可されたノードのリストを表示しています。
- （スキーマにより許可されている場合）選択されたノード、または、削除することのできる祖先要素を削除します。（スキーマに従い）削除される可能性のあるノードはサブメニューにリストされています。
- エンティティとCDATA セクションを挿入します。ドキュメントのために宣言されているエンティティはサブメニュー内にリストされています。CDATA セクションはテキスト内にのみ挿入することができます。
- ドキュメントのコンテンツの切り取り、コピー、（XML 、または、テキストとしての貼り付けを含む）貼り付け、および削除。

メモ インターフェイスに関する詳細は、次を参照してください: Authentic View インターフェイス 415

11.1.3 ノードを操作する

Authentic View XML ドキュメント内で発生するノードには2 つの主要な種類があります: **要素ノード** と**属性ノード**。これらのノード はオンに切り替えることのできる 405 タグによりマークアップされます。（マークアップされていない）テキストノードと周りのテキストから区別するためにマークアップされている）CDATA セクションノード などドキュメント内には他のノードも存在します。

このセクションで説明されたノードの操作は要素ノードと属性ノードのみを参照します。このセクション内で説明されたオペレーションを試す場合、大きなマークアップがオンに切り替えられていることが 405 勧められます。

メモ **同じレベル、または、ハイレベルの要素** のみが選択された要素の前、または、後に挿入されることに注意してください。同じレベルの要素は**兄弟**です。段落要素の兄弟は、他の段落要素ですが、リスト、テーブル、イメージであることができます。兄弟は要素の前、または、後に発生することができます。 ハイレベルの要素は **祖先** 要素と祖先の兄弟です。段落要素に関しては、祖先要素はセクション、チャプター、アーティクル、などである場合があります。有効な XML ファイル内の段落には既に祖先が存在します。結果として、Authentic View 内に高いレベルの要素を追加すると、関連する祖先の兄弟として新規の要素が作成されます。例えば、セクション 要素が段落の後に挿入されると、現在の段落要素を含むセクションの兄弟として作成されます。

ノードの操作の実行

ノードの操作は、コンテキストメニュー内でコマンドを選択、または、要素入力ヘルパー[406]内でノード操作エントリをクリックすることで行うことができます。一部の場合、要素、または、属性は、ドキュメントのAuthentic View 内のノードリンクの追加[408]をクリックして追加することができます。段落またはリストアイテムとして定義されている要素の特別な場合は、このような要素内を作成する際、Enter キー[409]を押すと、この種類の新規の兄弟要素が作成されます。このセクションは要素の適用[409]、ノードの削除[409]、および要素をクリアする[409]メカニズムを使用して、ノードを作成、または、削除するかについて説明されています。

要素の挿入

要素は次の場所に挿入することができます:

- 要素ノード内のカーソルの場所に挿入することができます。コンテキストメニューの「**挿入**」コマンドのサブメニュー内にその場所に挿入することのできる要素がリストされます。要素入力ヘルパー内では、その場所に挿入することのできる要素は アイコンにより表示されます。`NanonullOrg.xml` ドキュメント内では、`para` 要素内にカーソルを置き、コンテキストメニューと要素入力ヘルパーを使用して `bold` と`italic` 要素を作成します。
- スキーマにより許可されている場合、選択された要素、または、その祖先の前、または、後に挿入することができます。必要とされる要素をロールアウトされるサブメニューから選択します。要素入力ヘルパー内では、選択された要素の前、または、後に挿入することのできる要素は および アイコンにより表示されます。要素入力ヘルパー内では、選択された要素の前後のみに要素を挿入することができます。祖先要素の前後に要素を挿入することはできません。`para` 要素と社員をリストするテーブル内にカーソルを置いてこのコマンドを試してください。

ノードリンクの追加

要素、または、属性がドキュメントデザイン内に含まれ、XMLドキュメント内で存在しない場合、ノードが指定されているドキュメント内の場所でノードリンクの追加が表示されます。このリンクを確認するために、テキスト、ロゴの場所を持つライン内で、`CompanyLogo` 要素内で `@href` ノードを選択し、(「**削除**」キーを押して) 削除します。編集された`CompanyLogo` 要素内で `add @href` リンクが表示されます(*下のスクリーンショット*)。リンクをクリックすると、`@href` ノードがXMLドキュメントに追加されます。`@href` ノードがこのように表示されるようにデザインを指定しているため、`@href` タグ内のテキストボックスは、はこのように表示されます。`@href` ノード(または、コンテンツ)の値を入力する必要があるので、テキスト `nanonull.gif` を入力します。

要素のコンテンツモデルが曖昧な場合、例えば、子要素のシーケンスを指定する場合、順序に決まりはありませんので `add...` が表示されます。ノード名が指定されていないことに注意してください。リンクをクリックすると、有効に挿入することのできる要素のリストがポップアップします。

メモ ノードリンクの追加はドキュメントテンプレート内に直接表示されます。コンテキストメニュー、または、要素入力ヘルパーに対応するエントリはありません。

Enter キーを使用して新規要素を作成する

要素がスタイルシート デザイナーにより段落、または、リストアイテムとして書式設定されている場合、このようなノード内で「Enter」キーを押すと、点在のノードの後にこのような種類の新規のノードが挿入されます。NanonullOrg.xml ドキュメント内でこのメカニズムを試すことができます。(終了タグの前の) para ノードの終わりに移動し、「Enter」を押すことで実行することができます。

要素の適用

(テキストと子要素の両方を含む)複合型 コンテンツの要素内では、テキスト コンテンツの一部を選択し、許可されている子要素を適用することができます。選択されたテキストは適用された要素のコンテンツになります。要素を適用するには、コンテキスト メニュー内で、「**適用**」を選択し、適用することのできる要素から選択します(選択されたテキストに要素が適用されない場合、「**適用**」コマンドはコンテキスト メニュー内に表示されません)。要素入力ヘルプ パー内では、選択のために適用することのできる要素は、 アイコンで表示されます。NanonullOrg.xml ドキュメント内で、複合型 コンテンツ para 要素内のテキストを選択し、bold と italic 要素の適用を試してください。

スタイルシート デザイナーが要素を適用するツール バーアイコンを作成する場合があります。NanonullOrg.xml ドキュメント内では、bold と italic 要素をアプリケーションの Authentic ツール バー内の太字と斜体アイコンをクリックすることで適用できます。

要素の削除

ドキュメントを無効にしない場合、ノードを削除することができます。ノードの削除はその全てのコンテンツを削除します。コンテキスト メニュー内の「**削除**」コマンドを使用してノードを削除することができます。「削除」コマンドがハイライトされると、選択されたノードからドキュメントのトップ レベルのノードまで削除する全てのノードを含むサブメニューがポップアップされます。削除するためにノード選択するには、カーソルをノード内に置く、または、ノード(または、ノードの一部)をハイライトします。要素入力ヘルプ パー内では、削除することのできるノードは アイコンで表されます。削除可能なノードは選択後に「**削除**」キーを押すことにより削除することができます。NanonullOrg.xml ドキュメント内で、上記のメカニズムを使用していくつかのノードを削除してください。**Ctrl+Z** を使用して変更を元に戻すことができます。

要素の作成

複合型 コンテンツ(テキストと要素の子)を持つ要素の子である要素ノードをクリアすることができます。ノードが選択されると、または、入力ポイントとしてノードにカーソルがノード内に置かれると、要素の全体をクリアすることができます。要素内のテキスト フラグメントはテキスト フラグメントをハイライトすることにより要素マークアップをクリアすることができます。選択を決定し、コンテキスト メニュー内の「**クリア**」を選択して、要素をクリアします。要素入力ヘルプ パー内では、特定の選択のためにクリアする要素は アイコン(挿入ポイント 選択)と アイコン(範囲の選択)により示されます。NanonullOrg.xml ドキュメント内で、(複合型 コンテンツを持つ) para の bold と italic 子要素を使用して、クリアメカニズムを試してください。

テーブルとテーブルの構造

Authentic View テーブルには2つの型が存在します:

- *SPS テーブル(静的と動的)。*スタイルシート デザイナーにより SPS テーブルの大まかな構造は決定されます。この広義な構造では、許可されている唯一の構造的な変更はコンテンツ優先のみです。例えば、新規の行を動的な SPS テーブルに追加することができます。
- *XML テーブル*は特定のノードのコンテンツを表示します。(個人の特定の情報などを含んだ) テーブルなど。スタイルシート デザイナーが XML テーブルとしてこのノードの作成を有効化している場合、テーブルの構造を決定し、コンテンツを編集することができます。XML テーブルはAuthentic View 内のテーブル 432 セクションで詳細に説明されています。

11.1.4 Authentic View 内にデータを挿入する

Authentic View メインウィンドウ内のXMLドキュメントに直接データが入力されます。追加して、属性のためにデータ(属性の値)を属性入力ヘルパーに挿入[411] することができます。データは(i) 直接テキストとして入力されます。または、(ii) 定義済みのテキストエントリにマップされるデータ入力デバイス内のオプションを選択することにより入力されます。

テキストコンテンツの追加

Authentic View のメインウィンドウ内にテキストとして直接入力することができます。コンテンツを挿入するために、カーソルをテキストを入力する箇所にポイントし、入力を開始します。クリップボードからテキストをコピーして、ドキュメントに貼り付けることもできます。コンテンツは、「Caps」と「削除」キーなどの通常の編集機能を使用して編集することができます。例えば、編集するテキストをハイライトして、「Caps」キーがオンのままで変わりのテキストを入力することができます。

例えば、企業の名前を変更するには、`Office` の`Name` フィールド内でカーソルを Nanonull の後にポイントし、Nanonull, Inc. から Nanonull USA, Inc. に名前を変更するためにを`USA` 入力します。

テキストを編集することができる場合、カーソルをポイントしてハイライトします。それ以外の場合ハイライトすることができません。アドレスブロック内の「Street」、「City」、または「State/Zip」などの(フィールドの値ではなく) フィールド名 を変更しようと試みてください。StyleVision Power Stylesheet から生成されたテキストはXML コンテンツではないため、このテキスト内にはカーソルをポイントすることができません。

特別文字とエンティティの挿入

データの挿入する場合、次のコンテンツの型は特定の方法で扱われます:

- *特殊文字は、XML マークアップ*(アンパサンド、アポストロフィ、より大記号 (>)、より小記号 (<)、および引用符) のために使用されます。これらの文字は内蔵のエンティティ[412] として使用することができ、エンティティの入力ヘルパー内の対応するエンティティをダブルクリックすることによりドキュメント内に入力することができます。(例えば、プログラムコードリスティング内では) CDATA セクション内に挿入することができます。CDATA セクションを入力するには、CDATA セクションを入力する場所を右クリックして、コンテキストメニューから「**CDATA セクションの挿入**」を選択します。XML プロセッサーはCDATA セクション内の全てのマークアップ文字を無視します。CDATA セクション内で特殊文字を使用する場合、エンティティの参照ではなくその文字を入力します。
- *キーボードを使用して入力できない特殊文字* はシステムの文字マップからコピーういてドキュメント内の必要とする場所に貼り付けます。
- *頻繁に使用される文字列* は、エンティティの入力ヘルパー内に表示される をエンティティとして定義[448] することができます。エンティティは[412] 必要な場所にカーソルをポイントとして、入力ヘルパーとして挿入することができます。テキスト文字列の値は つの場所に保管されるため、これは管理のために役にたちます。値を変更する場合、エンティティの定義を変更するだけで行うことができます。

メモ Authentic View 内でマークアップが非表示の場合、空の要素を簡単に見逃す可能性があります。空の要素を見逃すことを回避するため、大きなまたは小さなマークアップをオンに切り替える[405]。

上記のテキストコンテンツの各種類の使用法を試してください。

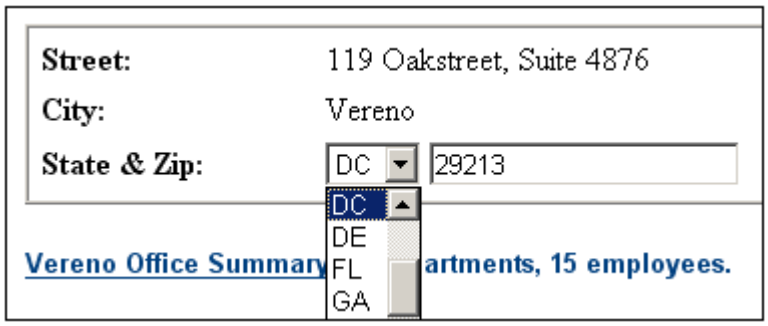
データ入力デバイスを使用してコンテンツを追加する方法

上記で学んだコンテンツの編集内で、テキスト内にコンテンツとして直接入力することにより追加することができます。**要素のコンテンツ**(または、属性の値) を以下の方法以外で Authentic View 内に入力することはできません。データ入力デバイスを使用して入力する

Authentic View 内のデータ入力デバイスのリストと各デバイスのための XML ファイルにどのようにデータが入力されるかの説明と共に下に示されています。

データ入力デバイス	XML ファイル内のデータ
入力フィールド（テキストボックス）	ユーザーにより入力されたテキスト
複数の入力フィールド	ユーザーにより入力されたテキスト
コンボボックス	値にマップされているユーザーの選択
チェックボックス	値にマップされているユーザーの選択
ラジオボタン	値にマップされているユーザーの選択
ボタン	値にマップされているユーザーの選択

アドレスフィールドを含む静的なテーブル内で、2 つのデータ入力デバイスがあります（*下に表示される*）： `Zip` フィールドの入力フィールドと州のフィールドのコンボボックス。テキストフィールド内に入力する値は、対応する要素の XML コンテンツとして直接入力することができます。他のデータ入力デバイスに関しては、選択された内容は値にマップされます。



上に示される Authentic View のための対応する XML テキストは以下のとおりです:

```
<Address>
  <ipo:street>119 Oakstreet, Suite 4876</ipo:street>
  <ipo:city>Vereno</ipo:city>
  <ipo:state>DC</ipo:state>
  <ipo:zip>29213</ipo:zip>
</Address>
```

コンボボックスの選択 `DC` は `DC` の値にマップされていることに注意してください。`Zip` フィールドの値は、`ipo:zip` 要素のコンテンツとして直接入力されます。

11.1.5 属性の値を入力する

属性は要素のプロパティで、要素は無制限の数の属性を持つことができます。属性には値があります。XML データを属性の値として入力する場合があります。Authentic View 内では、属性の値を2 つの方法で入力することができます:

- 属性が値を受け入れるように作成されている場合、メインウィンドウ内のコンテンツとして入力する
- 属性入力ヘルパー内で入力する

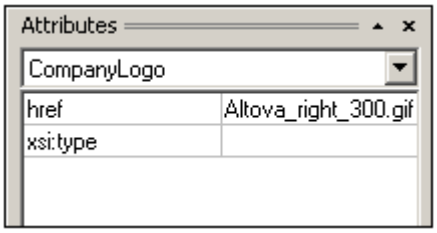
メインウィンドウ内の属性の値

属性の値をノーマルなテキスト、または、入力フィールド内にテキストとして、または、XML 値にマップされるユーザー選択として入力することができます。これらは要素のコンテンツが入力させる方法と同様に入力されます: 次を参照してください: Authentic View 内のデータの挿入 410。このような場合、要素のコンテンツと属性の値がStyleVision Power Stylesheet により区別され、データは適切に処理されます。

属性入力ヘルパー内の属性の値

属性の値を入力、または、変更する場合、属性入力ヘルパー内で行うこともできます。最初に、属性ノードをAuthentic View 内で選択すると、属性の値が属性ヘルパー内に入力、または、属性ヘルパー内で編集されます。`NanonullOrg.xml` ドキュメント内では、ロゴの場所は、`CompanyLogo` 要素の`href` 属性の値として保管されます。使用されるロゴを変更するには、以下を行います:

1. CompanyLogo タグをクリックして、CompanyLogo 要素を選択します。CompanyLogo 要素の属性は、属性入力ヘルパー内に表示されます。
2. 属性入力ヘルパー内で、href 属性 をnanonull.gif からAltova_right_300.gif に変更します(`Examples` フォルダー内のイメージ)。



Nanonull ロゴがAltova ロゴにより置き換えられます。

メモ 属性入力ヘルパー内にエンティティを入力することはできません。

11.1.6 エンティティを入力する

Authentic View 内のエンティティは、必ずしもそうではありませんが、通常、単一のテキスト文字列、およびXML ドキュメントのフラグメントのような XML データです。エンティティはイメージファイルのようなバイナリファイルであることができます。すべてのエンティティは特別のドキュメントのために使用することができ、エンティティの入力ヘルパー内で表示されます(*下のスクリーンショット*)。エンティティを入力するには、ドキュメント内の入力する箇所にカーソルをポイントし、エンティティ内のエンティティをダブルクリックします。属性入力ヘルパー内にエンティティを入力することはできないことに注意してください。

(アポストロフィ、より大記号 (>)、より小記号 (<) シンボル、および二重引用符と同様)アンパサンド文字 (&) はXML 内で特別な意味を有します。XML 特有の文字と区別するために、これらの文字を挿入するために、エンティティは使用されます。これらの文字はAuthentic View 内のエンティティとして使用することができます。

NanonullOrg.xml 内で、Joe Martin (Marketing) のタイトルを Marketing Manager Europe & Asia に変更してください。以下を行ってください:

1. アンパサンドが入力される箇所にカーソルをポイントしてください。
2. 「amp」としてリストされているエンティティをダブルクリックします。アンパサンドが入力されます (*下のスクリーンショット*)。

メモ エンティティの入力ヘルパーは状況に依存しません。全ての使用することのできるエンティティは、カーソルの位置に関係なく表示されます。これは、ドキュメント内の全ての場所にエンティティを入力できることを意味しません。わからない場合は、エンティティを入力した後、ドキュメントを検証してください:「XML | XML検証」(**F8**)。

自身のエンティティの定義

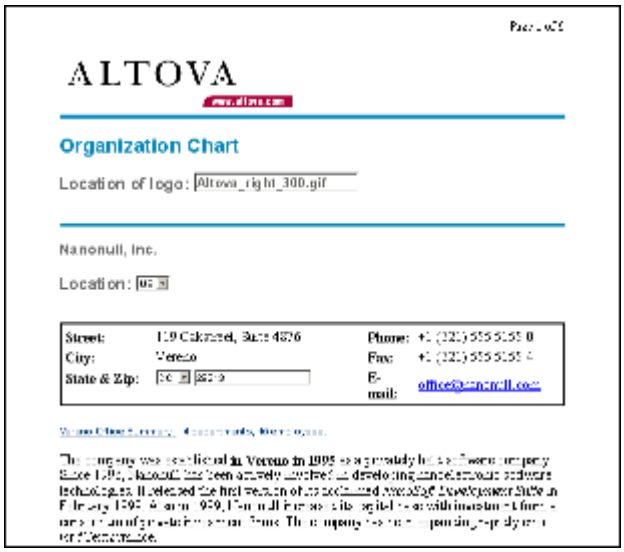
ドキュメントエディターとして、ドキュメントエンティティを定義することができます。この点についてセクション Authentic View 内でエンティティを定義する 448 内で詳しく説明されています。

11.1.7 ドキュメントを印刷する

XML ドキュメントの Authentic View からの印刷は、Authentic View 内の書式設定を保持します。

NanonullOrg.xml を印刷するには以下を行います:

1. まだ行っていない場合は、マークアップモードを非表示にするに切り替えます。マークアップを印刷しない場合は、これを行う必要があります。
2. 「ファイル | 印刷をプレビュー」を選択して、全てのページをプレビューします。50% に縮小された印刷のプレビューページが下に表示されています。ページの書式設定は Authentic View と同じであることに注意してください。



3. ファイルを印刷するには、「ファイル | 印刷」をクリックします。

マークアップを表示するドキュメントのバージョンを印刷することもできます。これを行うには、Authentic View を小さなマークアップモードの表示、または、大きなマークアップモードの表示に切り替えて印刷します。

11.2 Authentic View インターフェイス

Authentic View はアクティブなドキュメントのAuthentic タブをクリックすると有効化されます。SPS がXML ドキュメントに割り当てられていない場合、割り当てるようにプロンプトされます。「**Authentic | Stylevision スタイルシートを割り当てる**」コマンドを使用していつでもSPS を割り当てることができます。

このセクションでは以下について説明されています:

- インターフェイスの概要
- Authentic View に特有のツールバーの説明
- メインのAuthentic View ウィンドウで使用することのできるモードの表示に関する説明
- 入力ヘルパーの使用方法の説明
- コンテキストメニューの説明は、XML ドキュメント内のAuthentic View で見つけることができます。

Authentic View 情報の追加ソースは以下のとおりです:

- Authentic View チュートリアルはAuthentic View インターフェイスの使用方法を表示します。このチュートリアルはAltova XMLSpy とAltova Authentic Desktop 製品のドキュメント（チュートリアルのセクションを参照してください）、およびオンラインで見つけることができます。
- Authentic View メニューコマンドの詳細に関しては、次を参照してください: 製品ドキュメントのユーザーレファレンスセクション。

Altova Web サイト: XML コンテンツの編集、XML の作成

11.2.1 GUI の概要

Authentic View は、ウィンドウの上部のメニューバーとツールバー、および、残りのインターフェイスを占める次ぎの3 つの部分により構成されています: プロジェクトウィンドウ、メインウィンドウ、と入力ヘルパーウィンドウ。これは下に示されるとおりです。

メニューバー

メニューバー内で使用することのできるメニューは、製品ドキュメントのユーザーレファレンスセクション内で詳しく説明されています。

ツールバー

ツールバー内に表示されるシンボルとアイコンは、Authentic View ツールバーアイコン[416]のセクションで説明されています。

プロジェクトウィンドウ

XML、XSL、XML スキーマ、とエンティティファイルをプロジェクト内でグループ化することができます。プロジェクトファイルのリストを作成し変更するには、「**プロジェクト**」メニュー内のコマンドを使用します（この点に関しては、製品ドキュメントのユーザーレファレンスセクションで詳しく説明されています）。プロジェクトファイルのリストはプロジェクトウィンドウ内に表示されています。プロジェクトウィンドウ内のファイルは、ダブルクリックすることによりアクセスすることができます。

情報ウィンドウ

このウィンドウは現在 Authentic View 内で選択されているノードに関する情報を提供します。

メインウィンドウ

XML ドキュメントが表示され編集されるウィンドウです。セクション Authentic View メインウィンドウ[419]内で詳しく説明されています。

入力ヘルパー

次の3つの入力ヘルパーウィンドウがエリア内に存在します: 要素、属性、および、エンティティ。ウィンドウ（要素と属性 入力ヘルパー）内に表示される内容は状況に依存します。すなわち、ドキュメント カーソルが存在する位置により異なります。要素またはエンティティをドキュメントに入力ヘルパーをダブルクリックすることにより挿入することができます。属性入力ヘルパー内のその属性の値フィールドに属性の値が入力されます。詳細に関しては、次を参照してください: Authentic View 入力ヘルパー[421]

ステータスバー

ステータスバーは現在選択されているノードへのXPath を表示します。

コンテキストメニュー

メインウィンドウ内で右クリックすると表示されるメニューです。使用することのできるコマンドは、状況に依存した編集コマンドです。すなわち、選択されたノードに関連する構造とコンテンツを操作することができます。このような操作は、ノードの挿入、追加、または、削除、エンティティの追加、またはコンテンツの切り取りと貼り付けを含みます。

11.2.2 Authentic View ツールバーアイコン

Authentic View ツールバー内のアイコンは、コマンドのショートカットです。アイコンの一部は、Windows アプリケーション、または、Altova 製品で既に慣れ親しんだアイコンであり、その他は全く新しいアイコンです。このセクションでは、Authentic View の固有のアイコンについて説明されています。下の説明では、アイコンはグループ化されています。

XML マークアップの表示/非表示

Authentic View 内では、XML 要素、または、属性の全部（大きなマークアップ）または一部（小さなマークアップ）を名前と共に、または、名前をなしで表示することができます。4つのマークアップアイコンがツールバー内に表示され、**Authentic** メニュー内で対応するコマンドを使用できるようになります。

(C) 2015-2021 Altova GmbH

	マークアップを非表示にします。すべてのXML タグは既に折りたたまれたものを除いて非表示になります。(展開する通常の方法である) 「マークアップモードを非表示にする」内の折りたたまれたタグをクリックすると、ノードのコンテンツが表示されタグが非表示になります。
	小さなマークアップを表示します。XML 要素/属性 タグは名前と共に表示されます。
	大きなマークアップを表示します。XML 要素/属性 タグは名前無しで表示されます。
	複合型マークアップを表示します。StyleVision Power Stylesheet 内では、各 XML 要素、または、属性を表示のために(大きなマークアップ、または、小さなマークアップ)指定することができ、また、すべてを非表示にすることも出来ます。要素の一部がマークアップと共に表示、または、マークアップを隠して表示するように指定されているため、複合型マークアップモードと呼ばれます。複合型マークアップモード内では、Authentic View ユーザーはカスタム化されたマークアップを確認することができます。しかしながら、このカスタム化はStyleVision Power Stylesheet をデザインした個人により作成され、Authentic View ユーザーにより定義することはできません。

動的なテーブル構造の編集

動的な SPS テーブル 内の行はデータ構造の繰り返しです。各行は単一の要素の発生を表示しています。各行は、結果として、次と同じ XML サブ構造を持ちます。

動的な テーブルの編集コマンドは、動的な SPS テーブルの行を操作します。要素の発生の数と順序を変更することができます。しかしながら、個別の要素の発生のサブ構造の変更を意味するため、動的な SPS テーブルの列が編集されます。

動的な テーブル編集コマンドのためのアイコンはツール バー内に表示され、**Authentic** メニュー内でも使用できます。

	テーブルに行を追加する
	テーブルに行を挿入する
	現在のテーブル行を複製する(すなわち、セルにコンテンツが複製されます)
	現在の行を上に1つ移動する
	現在の行を下に1つ移動する
	現在の行を削除する

メモ これらのコマンドは、**動的な SPS テーブル** のみに適用されます。静的な SPS テーブル内で使用することはできません。Authentic View 内で使用されるテーブルの異なる種類は、このドキュメントのAuthentic View 内のテーブルの使用 432 セクション内で説明されています。

XML テーブルの作成と編集

データをテーブルとして表示するために、自身のテーブルを挿入することができます。このようなテーブルは、XML テーブルとして挿入されます。XML テーブルと書式 テーブルの構造を変更することができます。XML テーブルの作成と編集のためのアイコンは、下に表示されるようにツール バー内で使用することができます。セクション XML テーブル編集アイコン437 で詳細が説明されています。

アイコンに対応するコマンドは、**メニューアイテムとしては使用することはできません**。XML テーブルを使用するには、この機能は有効化され、StyleVision Power Stylesheet 内で適切に構成されている必要があります。

Authentic View 内で使用されているテーブルの型とXML テーブルの作成方法、及び、編集方法は Authentic View 内のテーブルの使用 432 内で説明されています。

テキストのフォーマット アイコン

Authentic View 内のテキストは、XML 要素を適用して、または、必要とされる書式設定を持つ属性を適用することにより書式設定することができます。このような書式設定が定義されると、StyleVision Power Stylesheet のデザイナーは、書式設定を適用するために Authentic View ツール バー内でアイコンを提供します。テキストのフォーマット アイコンを使用してテキストのフォーマットを適用するには、書式設定するテキストをハイライトして、適切なアイコンをクリックします。

DB 行 のナビゲーションアイコン

矢印アイコンは(左側から右側) 最初のレコードに移動する、前のレコードに移動する、レコード # ダイアログに移動する、次のレコードに移動する、最後のレコードに移動する、です。

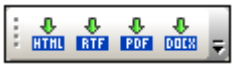
	このアイコンはクエリを入力することのできる「データベースクエリの変更」ダイアログを開きます。 Authentic View はクエリされたレコードを表示します。

XML データベースの編集

「**編集のための XML データを使用して新規の行を選択する**」コマンドにより、IBM DB2 などのXML DB 内の関連するテーブルから新規の行を選択することができます。この行は、Authentic View 内で表示され、編集することができ、DB に戻し保存することができます。

ポータブルな XML 書式 (PXF) ツール バー ボタン

XMLSpy とAuthentic Desktop のAuthentic View で次のPXF ツール バー ボタンを使用することができます:



個別のボタンをクリックすると、HTML、RTF、PDF、およびまたはDocX 出力が生成されます。

これらのボタンはAuthentic View 内でPXF ファイルが開かれるとこれらのボタンを使用することができます。特定の出力書式のためにPXF ファイルがXSLT スタイルシートを含むように構成されている場合、個別のボタンは有効化されます。例えば、PXF ファイルがXSLT スタイルシートをHTML とRTF のために含むように構成されている場合、HTML とRTF 出力のためのツール バー ボタンがのみが有効化され、PDF とDocX (W ord 2007+) 出力のためのボタンは無効化されます。

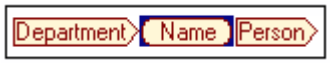
11.2.3 Authentic View メインウィンドウ

Authentic View には4つの表示モードが存在します: 大きなマークアップ、小さなマークアップ、複合型マークアップ、すべての マークアップを隠す。マークアップ 情報の多種のレベルをもつドキュメントをこれらのモードで確認することができます。モード間を切り替えるには、**Authentic** メニュー内のコマンド、または、ツールバー内のアイコンを使用します(前のセクションを参照してください: Authentic View ツールバーアイコン 416)。

大きなマークアップ

タグ内に要素/属性名を持つ要素と属性の開始と終了タグを表示します:

上の図の中では、要素 `Name` が**展開されており**、すなわち、開始と終了タグ、および要素のコンテンツが表示されています。要素/属性は、開始、または、終了タグをダブルクリックすることにより、**折りたたむことができます**。短くされた要素/属性を展開するには、短くされた タグをダブルクリックします。



大きなマークアップ内では、属性は属性の開始と終了タグ内で等号シンボルとして認識されます:

小さなマークアップ

名前の無い要素/属性の開始と終了タグを表示します :

Nanonull, Inc.

Location: US

Street: 119 Oakstreet, Suite 4876
City: Vereno
State & Zip: DC 29213

Phone: +1 (321) 555 5155 0
Fax: +1 (321) 555 5155 4
E-mail: office@nanonull.com

Vereno Office Summary: 4 departments, 15 employees.

The company was established **in Vereno in 1995** as a privately held software company. Since 1996, Nanonull has been actively involved in developing nanoelectronic software technologies. It released the first version of its acclaimed *NanoSoft Development Suite* in February 1999. Also in 1999, Nanonull increased its capital base with investment from a consortium of private investment firms. The company has been expanding rapidly ever since.

開始タグ内にはシンボルが存在し、終了タグは空であることに注意してください。また、要素タグには角かっこが使用されており、属性タグには等号サインがシンボルとして使用されていることに注意してください（*下のスクリーンショットを参照してください*）。

2006-04-01: Boston, USA

要素/属性を展開、または、折りたたむには、適切なタグをダブルクリックしてください。下のサンプルは、（青色でハイライトされている）折りたたまれた要素を表示しています。折りたたまれた要素と展開された要素の開始タグの形に注意してください。

Office Summary: 4 departments, 15 employees.

複合型マークアップ

複合型マークアップは、カスタム化されたマークアップのレベルを表示しています。StyleVision Power Stylesheet をデザインした個人はドキュメント内の個々の要素/属性のために大きなマークアップ、小さなマークアップ、または、マークアップ無しを決定することができます。Authentic View ユーザーは、内でカスタム化されたマークアップを複合型マークアップ表示モードで確認することができます。

すべての マークアップを非表示にする

すべてのXML マークアップは各されています。Authentic View 内で確認することのできる書式設定は、印刷されたドキュメントの書式であるため、この表示モードはドキュメントのW YSIW YG 表示です。

コンテンツの表示

Authentic View 内では、コンテンツは2つの方法で表示されています:

- テキスト形式。テキストを入力し、このテキストが要素のコンテンツ、または、属性の値になります。

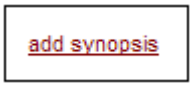
- データ入力デバイス。表示は、入力フィールド（テキストボックス）、複数の入力フィールド、コンボボックス、チェックボックス、または、ラジオボタンのいずれかを含んでいます。入力フィールドと複数の入力フィールドの場合、フィールドに入力するテキストが要素の XML コンテンツ、または、属性の値になります。

データ入力デバイスの場合、選択は StyleVision Power Stylesheet 内で指定される対応する XML 値を生成します。

このため、コンボボックス内では、（コンボボックスのドロップダウンリスト内で使用することができる）「許可された選択」選択は「1」の XML 値、または、「許可された選択」、またはその他をマップすることができます。「許可されていない選択」は「0」、「許可されていない選択」、または、その他にマップすることができます。

任意のノード

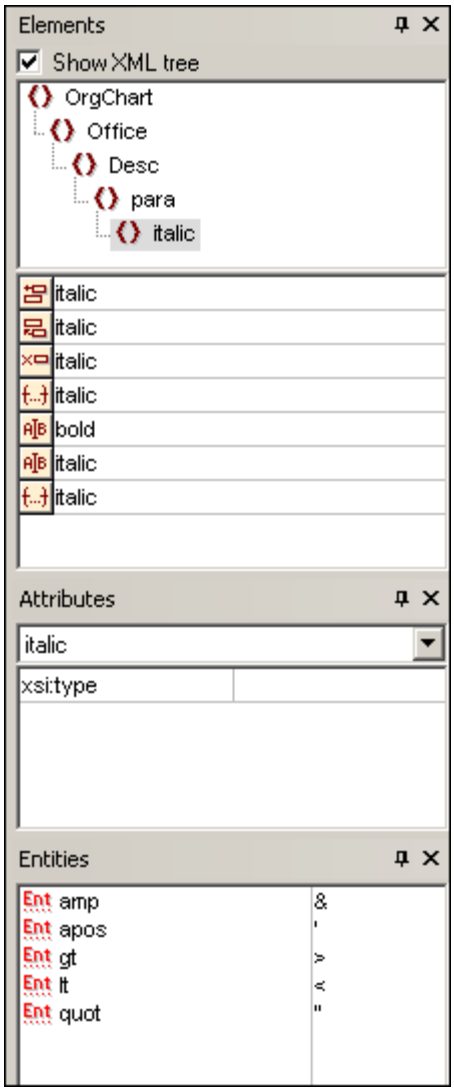
（参照されたスキーマに従い）要素、または、属性が **任意** の場合、型 `add [element/attribute]]` のプロンプトが表示されます:



プロンプトをクリックすると、要素が追加され、カーソルがデータの入力のために表示されます。複数の任意のノードが存在する場合、プロンプト `add...` が表示されます。プロンプトをクリックすると、任意のノードのメニューが表示されます。

11.2.4 Authentic View エントリヘルパー

Authentic View 内には3つの入力ヘルパーが存在します: 要素、属性、および、エンティティ。Authentic View インターフェイスの右下のウィンドウとしてこれらは表示されています（*下のスクリーンショットを参照してください*）。



要素と属性 入力ヘルパーは状況に依存しています。すなわち、入力ヘルパー内に表示される内容は、ドキュメント内のカーソルの位置により異なります。エンティティの入力ヘルパー内に表示されているエンティティは、状況に依存しません。ドキュメントのために許可されている全てのエンティティはカーソルの場所に関わらず表示されます。

それぞれの入力ヘルパーの説明に関しては下で説明されています。

要素入力ヘルパー

要素入力ヘルパーは2つの部分から構成されています:

- 上の部分には、「XML ツリーを表示」チェックボックスを使用してオンとオフを切り替えることのできる XML ツリーが含まれています。XML ツリーは、現在の要素のドキュメントのルート要素までの祖先を表示しています。XML ツリー内の要素をクリックすると、その要素に対応する要素（このリストの次のアイテム内で説明されているとおり）は要素入力ヘルパーの下の部分に表示されます。
- 挿入することのできるノードのリストを含む下の部分は、Authentic View 内の選択された要素、または、テキストの範囲から挿入、削除またはクリアすることができます。入力ヘルパー内の要素名の左側のアイコンにより入力ヘルパー内にリストされる要素が表示されます。要素入力ヘルパー内に発生するアイコンは意味を説明する詳細と共に下にリストされます。

入力ヘルパーからノードを使用するには、アイコンをクリックします。

要素の後に挿入
入力ヘルパー内の要素は、選択された要素の後に挿入されます。正確な階層レベルに追加されている場合、. 例えば、//sect1/para 要素内にカーソルが存在する場合、sect1 要素を追加すると、新規のsect1 要素が//sect1/para の兄弟の後ではなく、para 要素の親であるsect1 要素の兄弟の後に追加されます。

要素の前に挿入
入力ヘルパー内の要素は選択された要素の前に挿入されます。要素の後に挿入 コマンドと同様、要素は正確な階層レベルとして挿入されます。

要素の削除
要素とコンテンツを削除します。

要素の挿入
入力ヘルパーからの要素を要素内に挿入することもできます。要素内にカーソルが置かれると、その要素の許可されている子要素を挿入することができます。許可されている子要素は要素のみのコンテンツモデルの一部コンテンツモデル、および、複合型 コンテンツモデル(テキストと子要素)となることができます。

テキストの範囲が選択されていると、許可されている子要素がテキスト内で入力ポイントとしてカーソルがポイントされている場所に挿入されます。

- テキストの範囲が選択され、要素が挿入されると、テキストの範囲は挿入された要素のコンテンツになります。
- 要素が入力ポイントに挿入されると、要素はそのポイントに挿入されます。

要素の挿入後、これらのインラインの要素のための表示される(要素入力ヘルパー内の) 要素アイコンをクリックすることによりクリアすることができます。テキストの範囲、または、入力ポイントとしてテキスト内にカーソルをポイントするかにより表示されるアイコンが決定されます(下を参照してください)。

要素の適用
(大きなマークアップの表示 ビュー内の開始、および終了タグをクリックすることにより)ドキュメント内の要素を選択する場合、および、要素が他の要素により置き換えられる場合 (例えば、para などの複合型 コンテンツ 要素内では、italic 要素 がbold 要素と置き換えられることができます)、このアイコンは入力ヘルパー内の要素が選択された(オリジナルの) 要素に適用することができることを示します。**要素の適用** コマンドを複合型 コンテンツの要素内のテキストの範囲に適用することもできます。テキストの範囲は適用された要素のコンテンツとして作成されます。

- 適用された要素が元の要素の子と**同じ名前を持つ子要素** を持つ場合、また、元の要素内でこの子要素のインスタンスが存在する場合、オリジナルの子要素が新規の要素のコンテンツ内で保持されます。
- 適用された要素が元の要素のインスタント化された子と**同じ名前を持たない子要素** を持つ場合、元の要素のインスタント化された子が新規の要素が持つ子要素、または、要素の兄弟として追加されます。
- 適用された要素が元の要素のコンテンツモデル内に**等価が存在しない子要素**を持つ場合、この子要素は直接作成されず、Authentic View が挿入するオプションを提供します。

要素ではなくテキストの範囲が選択されている場合、要素の選択への適用は、コンテンツとして選択されたテキストの範囲と共にその場所で適用された要素を作成します。カーソルが入力ポイントの場合要素の適用は許可されていません。

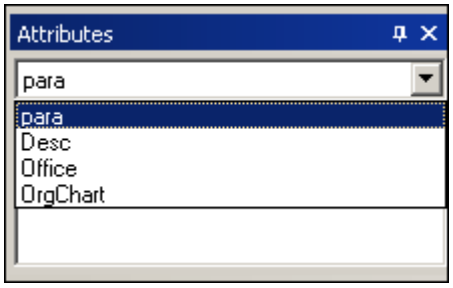
要素をクリアする
このアイコンは、複合型コンテンツの要素内のテキストが選択されると表示されます。アイコンをクリックすると選択されたテキストの範囲の周りから要素をクリアします。

要素をクリアする(挿入ポイントが選択されている場合)
複合型の子-コンテンツ 要素である要素内にカーソルが置かれるとこのアイコンが表示されます。アイコンをクリックするとインラインの要素がクリアされます。

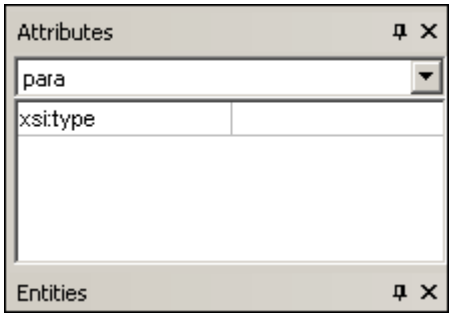
属性入力ヘルパー

属性入力ヘルパーはドロップダウンコンボボックスと属性のリストから構成されています。選択した要素がコンボボックス内に表示されます（開始、または、終了タグをクリック、または、要素コンテンツ内にカーソルをポイントして選択します）。

下に示されている属性入力ヘルパーにはコンボボックス内の `para` 要素が表示されています。コンボボックス内の矢印をクリックとすると、この場合 `OrgChart` である、`para` 要素の **祖先 ドキュメントのルート要素までの祖先** を表示するドロップダウンリストが表示されます。



コンボボックスの下に、要素のための有効化属性のリストが表示されます。この場合、`para` のためのリストが表示されます。要素で属性が必須の場合、太字で表示されます。（下のサンプルでは、内蔵の属性 `xsi:type` 以外の必須の属性は存在しません）。



属性のために値を入力するには、属性の値フィールドをクリックして、値を入力します。これにより属性とその値がXMLドキュメント内に作成されます。

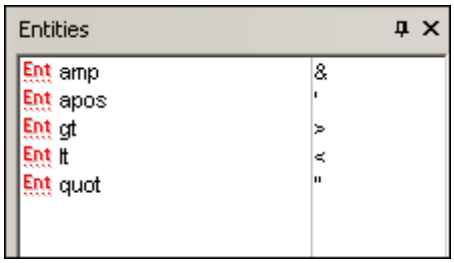
以下に注意してください：

- nillable 要素が選択された場合属性入力ヘルパー内に表示される `xsi:nil` 属性の場合、`xsi:nil` 属性の値は属性の値のためのドロップダウンリストから許可されている値（`true`、または、`false`）の1つを選択して入力することができます。
- `xsi:type` 属性を属性の値フィールド内をクリックして変更することができ、ドロップダウンリストからリストされた値を選択することができます。リストされた値は、Authentic View ドキュメントがベースとされるXMLスキーマ内で定義された使用することのできる抽象型です。

エンティティの入力ヘルパー

エンティティの入力ヘルパーによりドキュメントにエンティティを挿入することができます。エンティティを使用してドキュメントに（企業名など）頻繁に発生する特殊文字、または、テキストフラグメントを挿入することができます。エンティティを挿入するには、エンティティを挿入するテキストにカーソルを置き、エンティティの入力ヘルパー内のエンティティをダブルクリックします。

(C) 2015-2021 Altova GmbH



メモ 内部のエンティティはDTD 内で定義された値を持ちます。外部エンティティは他のXML ファイルなどの外部ソース内に含まれる値を持ちます。内部と外部エンティティはエンティティの入力ヘルパー内にリストされています。エンティティを挿入すると、– 値ではなく – 内部の、または、外部のエンティティがXML テキストに挿入されます。エンティティが内部のエンティティの場合、Authentic View **エンティティの値** を表示します。エンティティが外部エンティティの場合、Authentic View は値ではなくエンティティを表示します。これは、Authentic View 内で表示される外部エンティティであるXML ファイルを意味します。コンテンツは Authentic View 表示内のエンティティを置き換えません。

Authentic View 内で自身のエンティティを定義することができ、これらは入力ヘルパー内で表示されます: Authentic View セクション内の編集のエンティティの定義 448 を参照してください

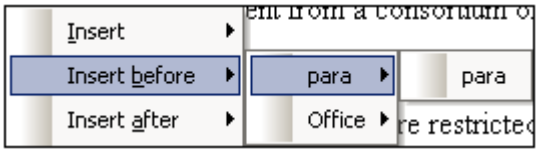
11.2.5 Authentic View コンテキストメニュー

選択されたドキュメントのコンテンツ、選択に関連したコマンドを使用して表示されるコンテキストメニュー、または、カーソルの場所を右クリックします。

要素の挿入

下の図は現在のカーソルの位置で挿入することのできる全ての要素リストである**サブメニューの挿入** を表示しています。**サブメニューの前に挿入** は現在の要素の前に挿入することのできる全ての要素をリストしています。**サブメニューの後に挿入** サブメニューは現在の要素の後に挿入することのできる全ての要素をリストしています。下の図では現在の要素は`para` 要素です。`bold` と`italic` 要素は、現在の`para` 要素内に挿入することができます。

下で確認することができるように、`para` と`Office` 要素は現在の`para` 要素の前に挿入することができます。



ノードの挿入、置換え、(**適用**)、およびマークアップの削除 (**クリア**) コマンドはコンテキストメニュー内、およびAuthentic View 入力ヘルパー 421 内で使用することができそれぞれのセクションで説明されています。

エンティティの挿入

「エンティティの挿入」コマンドにカーソルをポイントすると、宣言されている全てのエンティティのリスト含むサブメニューが表示されます。エンティティをクリックすると、選択された場所にエンティティが挿入されます。次を参照してください: ドキュメントのためにエンティティを定義する方法に関してはエンティティの定義 448 を参照してください。

CDATA セクションの挿入

このコマンドは、カーソルがテキスト内にポイントされると有効化されます。クリックするとカーソルの位置で CDATA セクションが挿入されます。CDATA セクションは開始と終了タグにより区切られています。大きな、小さなマークアップをオンにしてこれらのタグを確認することができます。CDATA セクション内では、XML マークアップと解析は無視されます。XML マークアップ文字(アンパサンド、アポストロフィ、より大記号(>)、より小記号(<)、および引用符) はマークアップとして扱われず、リテラルとして扱われます。ですから、CDATA セクションは、XML マークアップ文字を持つプログラムコードリスティングなどのテキストのために役にたちます。

ノードの削除

「**削除**」コマンドにマウスのカーソルをポイントすると、選択されたノードにより構成されたメニューリストとドキュメントを無効化しない)削除可能な祖先すべてが表示されます。削除する要素をクリックします。これは要素、または、削除可能な祖先を簡単に削除する方法です。祖先要素をクリックすると、選択されて要素を含む全ての子孫が削除されます。

クリア

「**クリア**」コマンドは要素マークアップを選択の周りからクリアします。ノード全体が選択されると、要素 マークアップはノード全体のためにクリアされます。テキストセグメントが選択されると、要素 マークアップがそのテキストセグメントのみのためにクリアされます。

適用

「**適用**」コマンドはメインウィンドウ内の選択の選択された要素に適用されます。詳細に関しては、次を参照してください: Authentic View 入力ヘルパー 421。

コピー、切り取り、貼り付け

これらは標準ウィンドウコマンドですが「**貼り付け**」コマンドはコピーされたテキストを SPS 全体のために指定されたスタイルシートのデザイナーにより XML、または、テキストとして貼り付けます。「**XML としてコピー**」と「**テキストとしてコピー**」コマンドの詳細に関しては、次を参照してください: 下記の「**貼り付け**」コマンドの詳細。

貼り付け

「**貼り付け**」コマンドは Authentic View XML フラグメントに(クリップボードにコピーされた内容を) XML、または、テキストとして張り付けるオプションを提供します。コピーされたフラグメントが XML として貼り付けられると、XML マークアップと共に貼り付けられます。テキストとして貼り付けられると、(XML マークアップではなく)コピーされたフラグメントのテキストコンテンツのみが貼り付けられます。次のシチュエーションが可能です:

- *マークアップタグと共にノード全体* が Authentic View 内でハイライトされ、クリップボードにコピーされます。(i) ノードはこのノードが有効な場所で XML として解析されます。無効な場所では解析されません(ii) ノードがテキストとして解析されると、(マークアップではなく)ノードの テキストコンテンツのみが貼り付けられます。テキストを貼り付けることのできる XML ドキュメント内にテキストコンテンツを貼り付けることができます。
- *テキストフラグメント* が Authentic View 内でハイライトされ、クリップボードにコピーされます。(i)フラグメントが貼り付けられる場所で XML ノードが有効な場合のみ、このフラグメントが XML として貼り付けられる場合、テキストの XML マークアップタグは、テキストフラグメントと共に明示的にコピーされず、テキストと共に貼り付けられます。(ii) フラグメントがテキストとして貼り付けられる場合、テキストが貼り付けられる XML ドキュメント内の場所に貼り付けられます。

メモ テキストは、テキストが許可されているノードにコピーされます。コピーされたテキストがドキュメントを無効化しないように注意してください。コピーされたテキストは以下のようになります:

(i) 新規の場所で構文的に有効な場合(例えば、数字のノード内の数字以外の文字 が無効になります)。
(ii) 以外はノードを無効化する場合(例えば、3 つの桁のみ受け入れるノード内の 4 つの桁はノードを無効化します)。

メモ 解析されたテキストがドキュメントを無効化する場合、テキストが赤い色で表示されます。

削除

「**削除**」コマンドは選択されたノードとそのコンテンツを削除します。ノード内にカーソルをポイント、または、ノードの開始、または、終了タグをクリックすることによりノードは選択されているとして考えられます。

11.3 Authentic View 内で編集する

このセクションはAuthentic View の重要な機能について詳しく説明しています。機能は頻繁に使用されるため、またはは、メカニズム、または、概念について説明が必要なため、これらの機能はこのセクションに含まれています。

セクションでは以下について説明されています:

- Authentic View 内で使用されているテーブルには3つの種類が存在します。セクション Authentic View 内のテーブルを使用する 432 はテーブルの3つの種類（静的な SPS、動的な SPS、および XML）がいつどのように使用されるかについて説明しています。最初に広義および概念についての説明がされ、使用方法の詳細について説明されます。
- 日付の選択は、日付をクリックすると、正確な XML 書式で日付を入力する視覚的なカレンダーです。次を参照してください: 日付の選択 446 。
- エンティティは、特殊文字、または、テキスト文字列の短縮形です。これらの特殊文字 、または、テキスト文字列を挿入するために、対応するエンティティを挿入することにより、自身のエンティティを定義することができます。詳細に関しては、次を参照してください: エンティティの定義 448
- Altova 製品のEnterprise とProfessional エディションでは、Authentic View ユーザーは、デジタルXML 署名 449 を使用してXMLドキュメントを署名し、署名を検証することができます。
- Authentic View で表示することのできるイメージ 450 。

Altova W eb サイト: XML コンテンツの編集、XML の作成

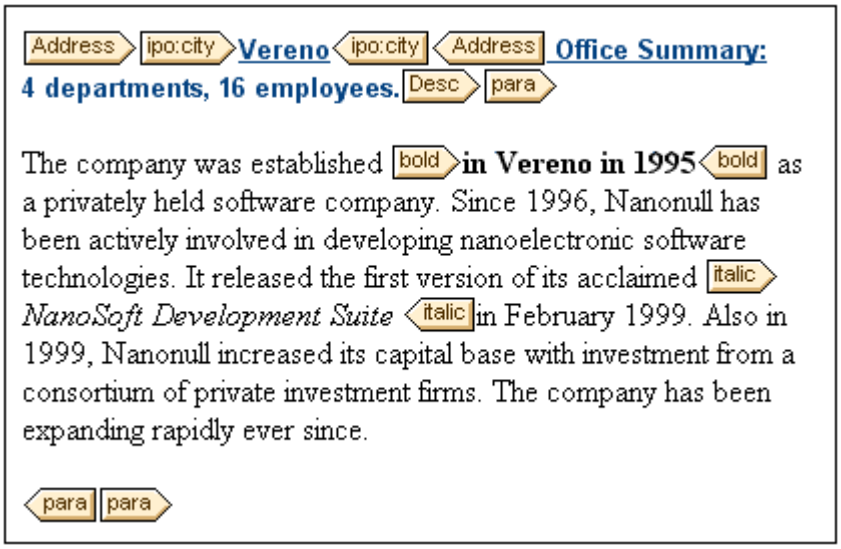
11.3.1 基本的な編集

Authentic View 内で編集を行う場合、XMLドキュメントが編集されます。Authentic View は、ドキュメントの構造的な XML マークアップを非表示にすることができ、ドキュメントのコンテンツのみを表示することができます（*下最初のスクリーンショット*）。 XML の技術的な側面に触れることなく、ドキュメントをノーマルなテキストドキュメントと同様に編集することができます希望する場合、編集中にマークアップをオンに切り替えることもできます（*下の2番目のスクリーンショット*）。

Vereno Office Summary: 4 departments, 16 employees.

The company was established **in Vereno in 1995** as a privately held software company. Since 1996, Nanonull has been actively involved in developing nanoelectronic software technologies. It released the first version of its acclaimed *NanoSoft Development Suite* in February 1999. Also in 1999, Nanonull increased its capital base with investment from a consortium of private investment firms. The company has been expanding rapidly ever since.

XML マークアップの存在しない編集することのできる Authentic View ドキュメント

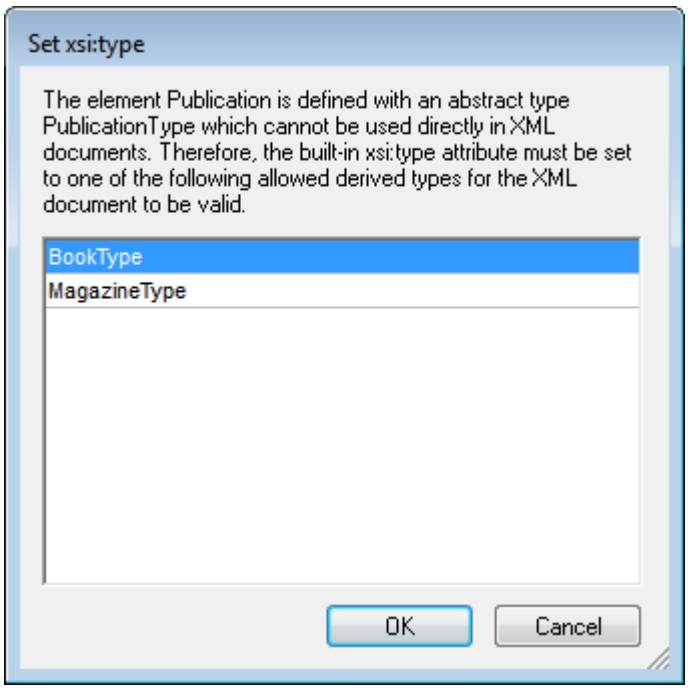


XML マークアップタグが存在する編集することのできる Authentic View ドキュメント

ノードの挿入

新規のノードをAuthentic XMLドキュメントに追加する必要がある場合があります。例えば、新規の`Person`要素はドキュメントのアドレスブックの型に追加される必要があります。このような場合、XML スキーマは新規の要素の追加を許可します。新規のノードを追加する箇所の後にAuthentic Viewドキュメント内でノードをクリックするだけです。表示されるコンテキストメニュー内で、必要に応じて「**前に挿入**」、または、「**後に挿入**」を選択します。ドキュメントに挿入することのできるノードはサブメニュー内にリストされます。必要とされるノードをクリックして挿入します。全ての必須の子孫ノードも挿入されます。子孫ノードが任意の場合、クリックすることのできるリンク`Add NodeName`が表示され、任意のノードを追加することができます。

追加されるノードが抽象型の要素の場合、(*下のスクリーンショットで表示されるような*) XML スキーマ内で使用することのできる生成された型のリストを含むダイアログが表示されます。



Publication 要素が追加されると、上のスクリーンショットが表示されます。Publication 要素は抽象的な複合型である型 PublicationType です。2つの複合型 BookType とMagazineType は抽象型 PublicationType から派生します。結果として、Publication 要素がXMLドキュメントに追加されると、Publication の抽象型から派生した2つの具体型の1つが指定される必要があります。新規のPublication 要素はxsi:type 属性に追加されます:

```
<Publication xsi:type="BookType"> ... </Publication>
<Publication xsi:type="MagazineType"> ... </Publication>
...
<Publication xsi:type="MagazineType"> ... </Publication>
```

使用することのできる生成された型の1つを選択し、「OK」をクリックすると以下を行うことができます:

- 選択された生成された型を要素のxsi:type 属性の値として設定します。
- 選択された生成された型のコンテンツモデル内で定義された子孫ノードと共に要素を挿入します。

属性入力ヘルパー内の要素のxsi:type 属性の値を変更することにより、選択された生成された型は後で変更することができます。要素の型がこのように変更されると、前の型のコンテンツモデルのノードは、削除され新しいコンテンツモデルのノードが挿入されます。

テキストの編集

Authentic View ドキュメントは、基本的にテキストとイメージにより構成されています。ドキュメント内のテキストを編集するには、テキストを挿入する箇所にカーソルを置き、編集します。(「**削除**」キーなどの)キーストロークとドラッグアンドドロップ機能を使用して、テキストのコピー、移動、削除を行うことができます。「**Enter**」キーは唯一の例外です。Authentic View ドキュメントは書式設定済みのため、追加のラインやスペースをアイテムの間に追加することができません。Authentic View 内の「**Enter**」キーは、このため、現在編集されている要素の他のインスタンスに追加するためにしようされ、この目的のみに使用することができます。

XML またはテキストとしてコピーする

XML、または、テキストとしてコピーし貼り付けることができます。

- テキストがXML として解析されると、XML マークアップは、ノードのテキスト コンテンツと共に解析されます。XML マークアップは、ノードのコンテンツの一部がコピーされた場合でも、貼り付けられます。マークアップを貼り付けるには、貼り付ける場所でスキーマに従い、貼り付けが許可されている必要があります。
- テキストがテキストとして解析されると、XML マークアップ破壊席されません。

XML 、または、テキストとして貼り付けるには、テキスト(**Ctrl+C**)をコピーし、テキストを貼り付ける箇所を右クリックし、コンテキストメニューコマンド「**貼り付け | XML**」、または、「**貼り付け | テキスト**」. を選択します。ショートカット「**Ctrl+V**」が使用される場合、テキストは、デフォルトのSPS のデフォルトの貼り付けモードで貼り付けられます。SPS のデザイナーにより貼り付けモードは指定することができます。詳細に関しては、次を参照してください: セクション コンテキストメニュー 425.

更に、ハイライトされテキストを貼り付ける箇所にドラッグします。テキストがドロップされるとポップアップが表示され、テキストをテキストまたは、XML として貼り付けるかが問われます。希望するオプションを選択します。

テキストのフォーマット

XMLドキュメントのシステムの基本的な概念は、プレゼンテーションとコンテンツは区別されるということです。XMLドキュメントにはコンテンツが含まれますが、スタイルシートにはプレゼンテーション(書式設定)が含まれています。Authentic View 内では、XMLドキュメントはスタイルシートにより表示されています。Authentic View 内で表示されるすべての書式設定はスタイルシートにより作成されていることを意味します。太字のテキストが表示されると、スタイルシートにより太字の書式設定が提供されます。リスト書式、または、テーブル書式を表示するリスト、または、テーブルはスタイルシートにより提供されています。Authentic View 内で編集するXMLドキュメントにはコンテンツのみが含まれており、書式設定は含まれていません。スタイルシートに書式設定が含まれています。Authentic View ユーザーは編集するテキストの書式設定をする必要が無い、または、できないことを意味します。コンテンツのみが編集されます。コンテンツに自動的に適用される書式設定は、データのセマンティクスおよび/または構造的な値にリンクされています。例えば、(セマンティックな単位として考えることのできる)電子メールアドレスは電子メールであるため、特定の方法で自動的に書式設定されます。同様に、ヘッドラインもドキュメント(構造的な及び構文的

なユニットを含む内の特定の場所で発生する必要があり、スタイルシートデザイナーがそのヘッドラインのために指定する書式に自動的に書式設定されます。電子メールアドレス、または、ヘッドラインの書式設定を変更することができます。電子メールアドレス、または、ヘッドラインのコンテンツを編集するだけです。

一部の場合、コンテンツは特別に表示される必要があります。例えば、テキスト文字列を太字で表示するなど。すべてのこのような場合、プレゼンテーションはドキュメントの構成要素に結び付けられている必要があります。例えば、表示するテキスト文字を太字で表示するなど、太字で表示することのできるスタイルシートデザイナーによるマークアップにより周りのコンテンツから構造的に区別される必要があります。Authentic View ユーザーとして、このようなテキスト文字列を使用する場合テキスト文字列を適切な要素マークアップで囲む必要があります。これを行うための情報に関しては、次を参照してください: ドキュメントの要素入力ヘルパー[422] セクション内の要素の挿入 コマンド

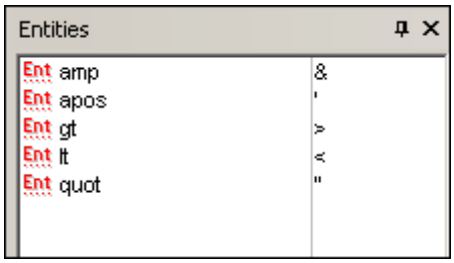
Authentic View 内で RichEdit を使用する

Authentic View 内で、RichEdit ツールバーとして作成された要素の中にカーソルが置かれている場合、ボタンとコントロールは有効化されます(*下のスクリーンショット*)。それ以外の場合、グレーアウトされます。

RichEdit ツールバーのボタンとコントロールを使用して適用するスタイルを指定し、スタイルするテキストを選択します。RichEdit により Authentic View ユーザーはフォント、フォントの大きさ、フォントのスタイル、フォントの飾り、色、テキストの背景色と配置を指定することができます。スタイルされたテキストはスタイル要素のタグに囲まれています。

エンティティの挿入

XML ドキュメント内では、文字の一部はマークアップのために保管されており、ノーマルなテキストでは使用することができません。これらの文字はアンパサンド (&)、アポストロフィ(')、より小記号 (<)、より大記号 (>)、引用符 (") 文字です。これらの文字をデータ内で使用する場合、エンティティの入力ヘルパー[424] を使用して、エンティティレファレンスとして挿入します (*下のスクリーンショット*)。



XML はカスタムのエンティティを作成する機会を与えます。

これらは以下であることができます: (i) キーボードで使用することのできない特殊文字。(ii) ドキュメントのコンテンツ内で再利用するテキスト文字列。(iii) XML データフラグメント、または、(iv) イメージなどの他の リソース。Authentic View アプリケーション内でエンティティを自身で定義する[448] ことができます。定義後、これらのエンティティは エンティティの入力ヘルパー[424] 内に表示され、ドキュメントに挿入されます。

CDATA セクションの挿入

CDATA セクションは XML データとして XML パーサーが処理しない XML ドキュメント内のテキストのセクションです。エンティティレファレンスによる特殊文字の置換えを希望しない場合テキストの大きなセクションをエスケープするために使用することができます。例えば、マークアップタグを使用して再生成するプログラムコード、または、XML フラグメントです。要素のコンテンツ内で CDATA セクションは発生することができ、開始と終了で `<![CDATA[` and `]]>` を使用してそれぞれ区切ることができます。この結果、テキスト文字列 `]]>` はセクションの終わりを完了前に示すため CDATA セクションは発生しません。この場合、より大きい文字はエンティティレファレンス(`>`) によりエスケープされるべきです。要素内に CDATA セクションを入力するには、希望する場所にカーソルを置き、右クリックし、コンテキストメニューから CDATA セ

ケションの挿入を選択します。CDATA セクションタグを Authentic View 内で確認するには、マークアップの表示オンに切り替えます[416]。更に、 CDATA セクション内に含まれるテキストをハイライトし、**CDATA セクションの挿入** コマンドを選択します。

メモ CDATA セクションを入力フィールド(すなわち、テキストボックスと複数のテキストボックス)に挿入することはできません CDATA セクションは、Authentic View にテキストコンテンツコンポーネントとして表示される要素内にのみ入力することができます。

リンクの編集

ハイパーリンクは2つの部分から構成されています: リンクテキストとリンクのターゲット。リンクテキストをテキスト内をクリックして編集することにより編集することができます。リンクのターゲットを編集することはできません。((静的な ターゲットアドレス、または、ターゲットアドレスを XML ドキュメント内に含まれるデータを生成することにより)スタイルシートのデザイナーによりリンクのターゲットが設定されます。) Authentic View からは、「Ctrl」を押してをクリックしてリンクのターゲットに移動することができます。(留意点: リンクをクリックするだけで、リンクテキストを編集するためのセットアップを行うことができます)。

11.3.2 Authentic View 内のテーブル

3つのテーブルの種類は、2つのカテゴリに分けることができます: SPS テーブル(静的と動的) とCALS/HTML テーブル。

SPS テーブルには2つの種対があります: 静的と動的。XML ドキュメントがリンクされている StyleVision Power Stylesheet のデザイナーによりSPS テーブルはデザインされます。XML ドキュメント内にSPS テーブルを挿入することはできず、SPS テーブルフィールドにデータを挿入し、動的な SPS テーブルに行を追加、または、削除することができます。SPS テーブル[432] のセクションでは、はこれらのテーブルについて説明されています。

CALS/HTML テーブルは Authentic View のユーザーにより挿入されます。目的は、ドキュメント階層構造内で希望する場所にテーブルを挿入することです。CALS/HTML テーブル[433] とCALS/HTML テーブル編集アイコン[437] の編集機能に関しては、下で説明されています。

11.3.2.1 SPS テーブル

Authentic View 内で2つの種類のSPS テーブルが使用されます: 静的なテーブルと動的なテーブル。

静的なテーブル

静的なテーブルは構造内およびセルのコンテンツ型内で固定されています。Authentic View のユーザーは、テーブルセルにデータを挿入することができますが、これらのテーブルの構造を変更(すなわち、行や列の追加)すること、セルのコンテンツ型の変更することはできません。テキスト内にタイプしてデータを入力、または、チェックボックス、または、ラジオボタンのフォーム内のオプションから選択して、または、コンボボックスのリストから選択してデータを入力することができます。データを入力すると編集することができます。

Nanonull, Inc.

Street:	119 Oakstreet, Suite 4876	**Phone:**	+1 (321) 555 5155
City:	Vereno	**Fax:**	+1 (321) 555 5155 - 9
State & Zip:	DC 29213	**E-mail:**	office@nanonull.com

メモ 動的なテーブルを編集するためのアイコン、または、コマンドは、静的なテーブルを使用する際に使用することはできません。

動的なテーブル

動的なテーブルには繰り返しデータ構造を表す行が存在します。すなわち、（静的なテーブルの場合は異なりますが）各行は同一のデータ構造を持ちます。結果として、行操作を行うことができます: 行の追加、行の挿入、行を上に移動する、行を下に移動する、行の削除。これらのコマンドは**Authentic** メニューとツール バー内のアイコンとして使用することができます（下に表示される）.

これらのコマンドを使用するには、適切な行内にカーソルをポイントし、必要とするコマンドを選択します。

Administration

First	Last	Title	Ext	EMail	Shares	Leave		
						Total	Used	Left
Vernon	**Callaby**	Office Manager	581	v.callaby@nanonull.com	1500	25	4	21
Frank	Further	Accounts Receivable	471	f.further@nanonull.com	0	22	2	20
Loby	Matise	Accounting Manager	963	l.matise@nanonull.com	add Shares	25	7	18
Employees: 3 (20% of Office, 9% of Company)					**Shares: 1500 (13% of Office, 6% of Company)**			
Non-Shareholders: Frank Further, Loby Matise.								

テーブル内のセルを移動するには、「上」、「下」、「右」、「左」矢印キーを使用します。前方の次のセルに移動する場合は、「タブ」キーを使用します。最後の行の最後のセル内の「タブ」キーを押すと、新規の行が作成されます。

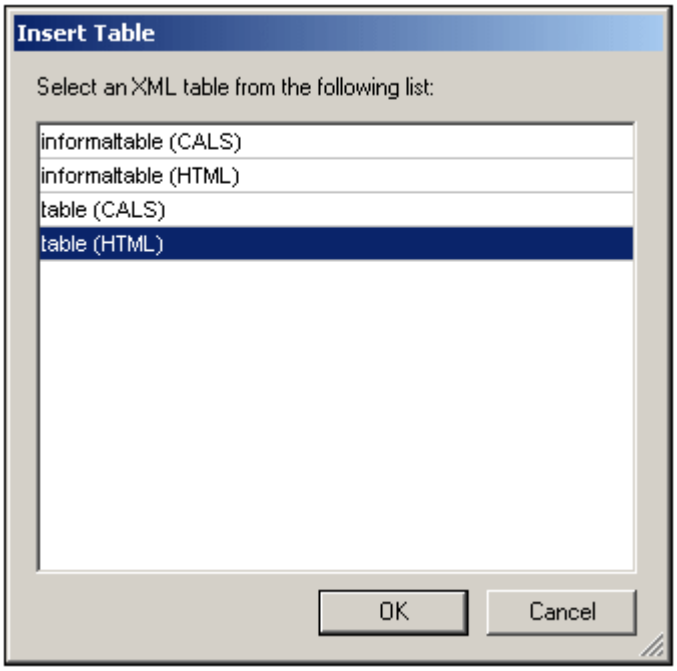
11.3.2.2 CALS/HTML テーブル

一部のXML データ構造がテーブル書式を表示するように指定されているため、Authentic View のユーザーはCALS/HTML テーブルを挿入することができます。CALS/HTML テーブルと作業するには3 つのステップが含まれています: テーブルの挿入。書式設定。データの挿入。CALS/HTML テーブルと作業するためのコマンドはツール バー内のアイコンとして使用することができます（次を参照してください: CALS/HTML テーブル編集アイコン 437）。

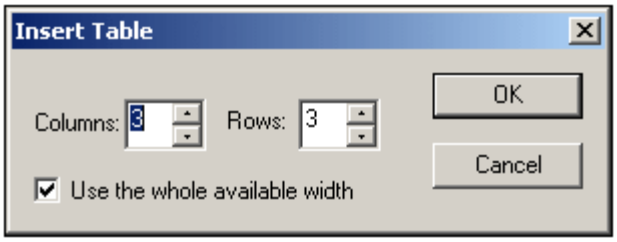
テーブルの挿入

CALS/HTML テーブルを挿入するには、以下を行います:

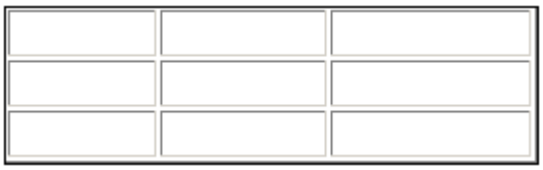
1. テーブルを挿入する箇所にカーソルをポイントし、アイコンをクリックします。（スキーマによりテーブルを挿入する箇所は決定されることに注意してください）。テーブルの挿入 ダイアログが表示されます（*下のスクリーンショット*）。このダイアログはテーブル構造が定義されているはすべてのXML 要素 データ構造をリストします。例えば、下のスクリーンショットではinformal テーブル要素とテーブル要素はCALS テーブル、および、HTML テーブルと定義されています。



2. 挿入する要素とテーブルモデルを含むエントリを選択し、「OK」をクリックします。
3. 次のダイアログ内では（*下のスクリーンショット*）列と行の数量を選択し、テーブルに追加されるヘッダーおよびまたはフッターを選択し、使用することのできる幅全てをテーブルが使用するかを指定します。完了すると、「OK」をクリックします。



上に示されるダイアログボックス内の仕様のために、以下のテーブルが作成されました。



テーブル メニューコマンドを使用して、列を追加、または、削除、列を作成、列をジョインし分割することができます。開始するには、広義の構造を作成する必要があります。

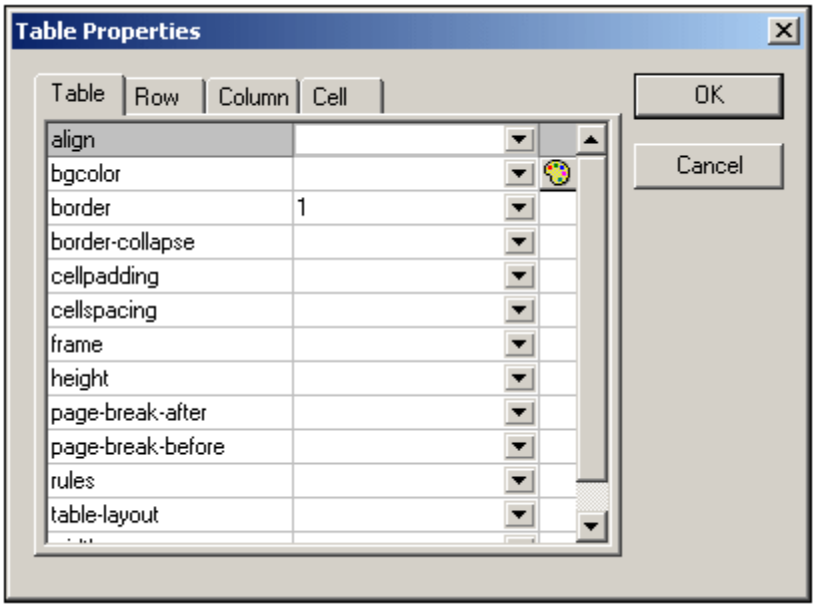
書式設定 テーブルとデータの挿入

テーブル書式設定はドキュメントデザインに既に割り当てられていますが、特定の環境ではテーブル書式設定を変更することができます。これらの環境は、以下のとおりです:

- 多種のテーブル構造要素に対応する要素は属性（基になる XML スキーマ内の）として定義されている適切な CALS、または、HTML テーブルプロパティを持つ必要があります。定義されている属性のみが書式設定のために使用できます。デザイン内で、値がこれらの属性のために設定されている場合、Authentic View 内でこれらの値をオーバーライドすることができます。
- デザイン内では、CSS スタイルを含むスタイル属性を設定してはなりません。CSS スタイルを持つスタイル属性が要素のために指定されている場合、スタイル属性その要素の他の書式設定の属性設定に対して優先順位が与えられます。 この結果、Authentic View 内で指定される書式設定は上書きされます。

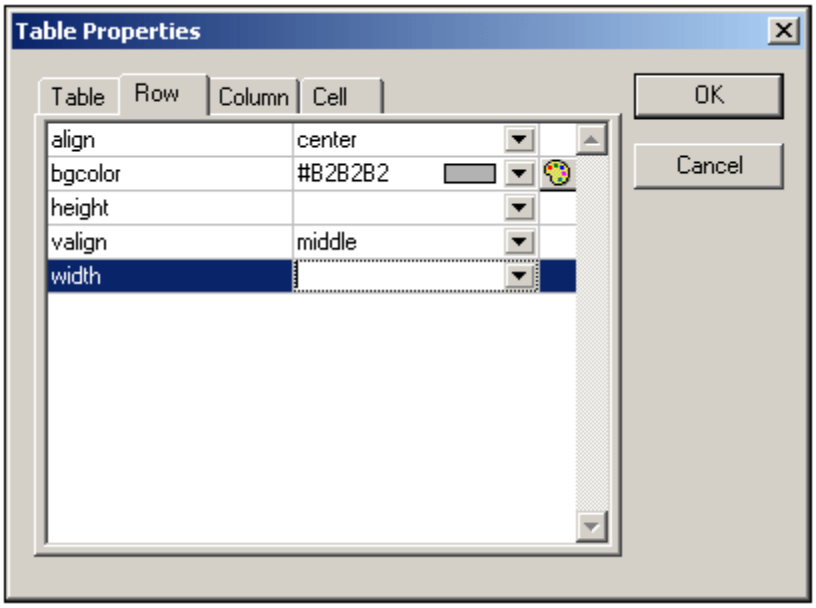
テーブル、行、列、または、セルの書式設定を行うには

1. テーブル （テーブルプロパティ）アイコンにカーソルを置きクリックします。テーブル、または、行、列、または、セルのための書式設定を行うテーブルプロパティダイアログが開かれます（*スクリーンショットを参照してください*）。



2. セルペースとセルパディングプロパティを「0」に設定します。テーブルは以下のようになります:

3. 最初の行にカーソルをポイントし、（テーブルプロパティ）アイコンをクリックします。「行」タブをクリックします。



最初の行がヘッダー行になるため、この行を他の行と区別するために背景色を設定します。上の図で行プロパティが設定されていることに注意してください。列ヘッダーテキストを挿入します。テーブルは以下のようになります:

Name	Telephone	Email

配置は指定されているとおり中央に揃えられています。

4. 「Telephone」列をサブ列「Office」と「Home」に分割する場合、「Telephone」列の幅を水平2つの列に分割する必要があります。最初にしかしながら、サブヘッダー行を作成するためにヘッダーセルの垂直に分割します。「Telephone」セル内にカーソルを置き、 （垂直に分割）アイコンをクリックします。テーブルは以下のようになります:

Name	Telephone	Email

5. 「Telephone」を含むセルの下のセルにカーソルを置き、 （水平に分割）アイコンをクリックします。列ヘッダー「Office」と「Home」を入力します。テーブルは以下のようになります:

Name	Telephone		Email
	Office	Home	

「Telephone」列内の各セルの幅を水平に分割します。

テーブル編集アイコンを使用して、列と行を追加、削除、垂直に整列することもできます。 CALS/HTML テーブル編集アイコンは、CALS/HTML テーブル編集アイコン437 内で説明されています。

テーブル内のセルを移動する

CALS/HTML テーブル内でセル間を移動するには、「上」、「下」、「右」、「左」矢印キーを使用します。

セルにデータを挿入する

セルにデータを入力するには、セル内にカーソルを置き、データを入力します。

テキストの書式設定

他のテキストXML ドキュメント内同様、CALS/HTML テーブル内のテキストは、 XML 要素、または、属性を使用して書式設定される必要があります。 要素を追加するには、 テキストをハイライトし、要素入力ヘルパー内の必要とされる要素をダブルクリックします。 属性の値を指定するには、テキストフラグメント内にカーソルを置き、属性入力ヘルパー内に必要とされる属性の値 を挿入します。書式設定 ヘッダーテキストを太字に設定すると、テーブルは以下のようになります。

Name	Telephone		Email
	Office	Home	

上のテキストはテキストのハイライトと太字がフォントの書式として指定されているグローバルテンプレートである要素 `strong` のダブルクリックにより書式設定されています。

メモ Authentic View 内で表示されるテキストの書式に関しては、必要とされるテキストのフォーマットを持つグローバルテンプレートが要素のためにStyleVision 内で作成される必要があります。

11.3.2.3 CALS/HTML テーブル編集アイコン

CALS/HTML テーブルを編集するために必要なコマンドは、ツールバー内のアイコンとして使用することができ、下にリストされています。これらのアイコンのために対応するメニューコマンドは存在しません。

CALS/HTML テーブルをいつどのように使用するかについての詳細は、CALS/HTML テーブル433 。

テーブルの挿入

「テーブルの挿入」コマンドは現在のカーソルの場所にCALS/HTML テーブルを挿入します。

テーブルの削除

「テーブルの削除」コマンドは現在アクティブなテーブルを削除します。

行の追加

「行の追加」コマンドは現在アクティブなテーブルの最後に行を追加します。

列の追加

「列の追加」コマンドは現在アクティブなテーブルの最後に列を追加します。

行の挿入

「行の挿入」コマンドは現在アクティブなテーブル内の現在のカーソルの場所の上に行を追加します。

列の追加

「列の追加」コマンドは現在アクティブなテーブル内の現在のカーソルの場所の左に列を挿入します。

セルを左側にジョインする

「セルを左側にジョインする」コマンドは、現在のセル(現在のカーソルの場所)を左側のセルとジョインします。両方のセルのタグは新規のセル内に留まり、列ヘッダーは変更されず、結合されません。

セルを右側にジョインする

「セルを右側にジョインする」コマンドは、現在のセル(現在のカーソルの場所)を右側のセルとジョインします。両方のセルのコンテンツは新規のセルに連結されます。

下のセルとジョインする

「下のセルとジョインする」コマンドは、現在のセル(現在のカーソルの場所)を下のセルとジョインします。両方のセルのコンテンツは新規のセルに連結されます。

上のセルとジョインする

「上のセルとジョインする」コマンドは、現在のセル(現在のカーソルの場所)を上のセルとジョインします。両方のセルのコンテンツは新規のセルに連結されます。

(C) 2015-2021 Altova GmbH

セルを左右に分割する

「セルを左右に分割する」コマンドは、現在アクティブなセルの右に新規のセルを作成します。両方のセルのサイズは元のセルと同じサイズです。

セルを上下に分割する

「セルを上下に分割する」コマンドは、新規のセルを現在アクティブなセルの下に作成します。

上揃え

このコマンドはセルコンテンツをセルの上に揃えます。

上下中央揃え

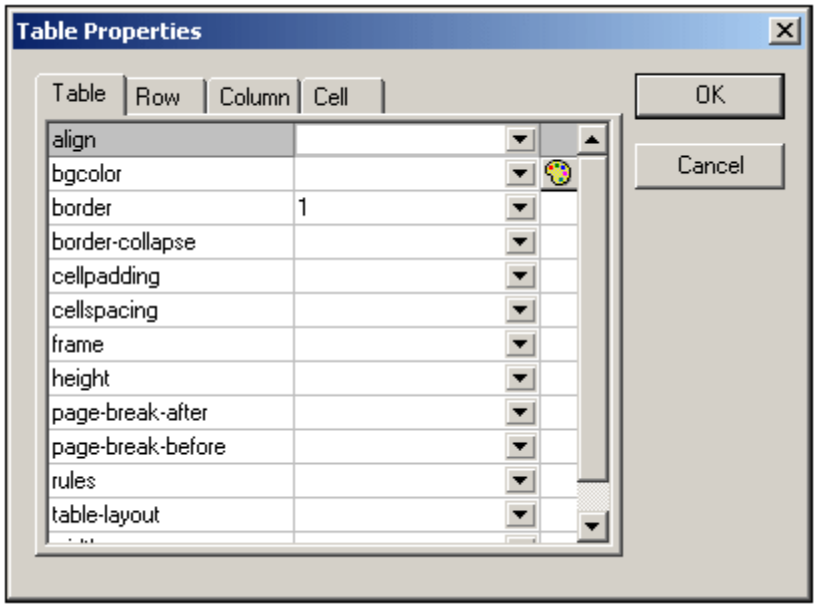
このコマンドは、セルコンテンツを上下中央に揃えます。

下揃え

このコマンドはセルコンテンツをセルの下に揃えます。

テーブルプロパティ

「テーブルプロパティ」コマンドは、テーブルプロパティダイアログボックスを開きます。このアイコンはHTML テーブルのためにのみ有効化することができますが、CALS テーブルのためにクリックすることはできません。



11.3.3 DB の編集

Authentic View 内で、データベース(DB) テーブルを編集し、データを DB に戻すことができます。このセクションには、DB テーブルを編集する際に使用することのできるインターフェイスの説明が含まれています。次の一般的な点について注意してください:

- デザインをコンパクトにまとめるために、Authentic View 内で表示される DB テーブル内のレコードの数量は StyleVision Power Stylesheet のデザイナーにより意図的に制限されている場合があります。このような場合、Authentic View にロードされるレコードの数量を制限することができます。DB テーブル行 ナビゲーションアイコンを使用して DB テーブル内に他のレコードをロードすることができます(*次を参照:* DB テーブルのナビゲート 440)。
- 特定のレコードを表示するために DB をクエリする 441 ことができます。
- DB レコードを追加、変更、削除し、DB にデータを戻すことができます。次を参照してください: DB テーブルの変更。 445

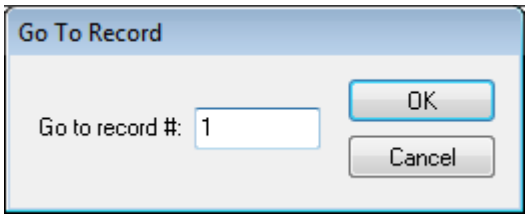
Authentic View 内で DB ベースの StyleVision Power Stylesheet を開くには、「**Authentic | データベースデータの編集**」をクリックして、必要とされる StyleVision Power Stylesheet を参照します。

メモ Authentic View 内では、SQLite データベースからのデータは編集することができません。Authentic View から SQLite データを保存しようと試みると、メッセージボックスにより既知の制限が通知されます。

11.3.3.1 DB テーブル内をナビゲートする

DB テーブル行をナビゲートするコマンドは、Authentic View ドキュメント内でボタンとして使用することができます。通常、4 – 5 つのボタンを持つナビゲーションパネルが各 DB テーブルに伴います。

矢印アイコンは、左側から右側、内の最初のレコードに移動する、前のレコードに移動する、レコードダイアログに移動する、次のレコードに移動する、最後のレコードに移動する、などがあります(*スクリーンショットを参照してください*)。



DB テーブルをナビゲートするには、必要なボタンを押してください。

XML データベース

IBM DB2 などのXML DB の場合、セル(または、行) には、単一のXMLドキュメントが含まれており、このため、単一の行が Authentic View にロードされます。他の行内のXMLドキュメントをロードする場合は、「Authentic | 編集のためにXML データを持つ新規の行を選択する」904 メニューコマンドを使用します。

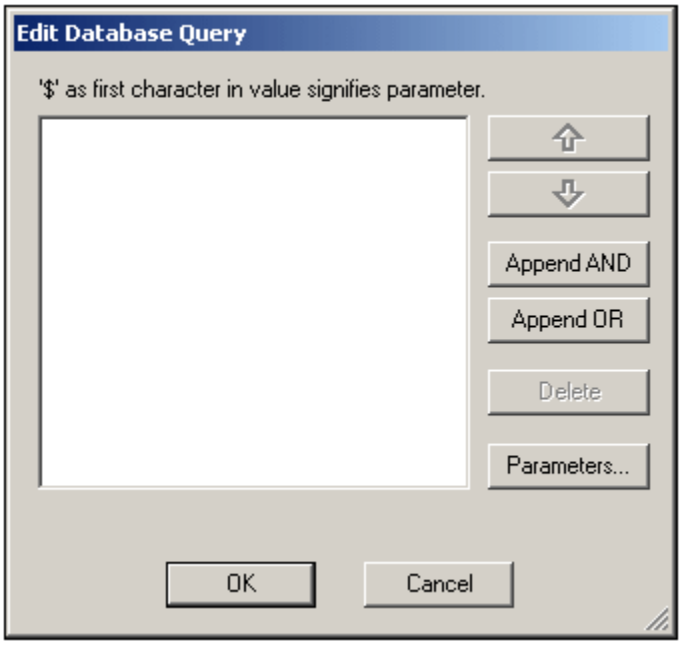
11.3.3.2 DB クエリ

DB クエリにより Authentic View 内に表示されるテーブルのレコードをクエリすることができます。クエリは個別のテーブルのために作成され、1つのテーブルに1つのクエリを作成することができます。クエリの送信時にAuthentic View ドキュメント内の変更が保存されない場合、ドキュメント内の全ての変更を保存、または、すべての変更を破棄するかが問われます。他のテーブル内の変更も保存/破棄されます。クエリの送信後、テーブルはクエリの条件を使用して再ロードされます。

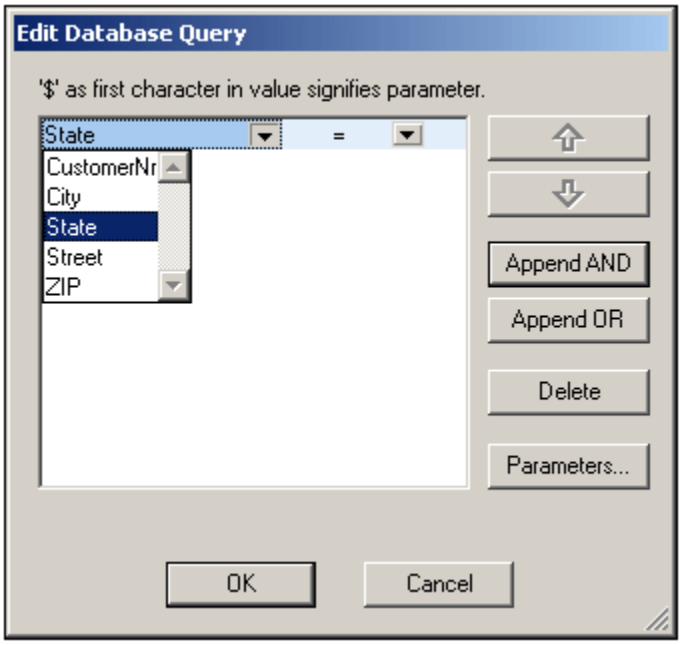
メモ 開かれているテーブルの数量が多すぎることを通知するメッセージが表示されると、開かれているテーブルの数量をクエリを使用してテーブル一部をフィルターし削減することができます。

クエリを作成して送信する方法:

1. データベースクエリの変更 ダイアログを開くために必要とされるテーブルのためのクエリボタン をクリックします(*スクリーンショットを参照してください*)。各 DB テーブルの上、または、下にこのボタンが表示されます。クエリボタンがテーブルに存在しない場合、StyleVision Power Stylesheet のデザイナーはそのテーブルに対して DB クエリ機能を有効化していません。



2. 「AND を追加」、または、「OR を追加」ボタンをクリックします。これにより（下に表示されるように）クエリのための空の条件が追加されます。



3. 条件のために式を入力します。式は以下により構成されています: (i)（関連付けられたコンボボックスで使用することのできる）フィールド名、(ii)（関連付けられたコンボボックスで使用することができます）演算子、(iii)（直接入力することのできる）値。式の構築方法に関しては、必要条件内の式 442 セクションを参照してください。
4. 他の必要条件を追加する場合、2つの条件をジョインするために使用するロジカルな演算子（AND、または、OR）に従い、「AND を追加」、または、「OR を追加」ボタンをクリックします。新規の必要条件を追加します。ロジカルな演算子の詳細に関しては、次を参照してください: セクション DB クエリ内の必要条件を再度並べ替える 443 。

条件内の式

DB クエリの必要条件内の式はフィールド名、演算子、および値により構成されています。**使用することのできるフィールド名** は、選択されたトップレベルのデータテーブルの子要素です。これらのフィールドの名前は、コンボボックス内にリストされます（*上のスクリーンショットを参照してください*）。「**演算子**」は下にリストされています:

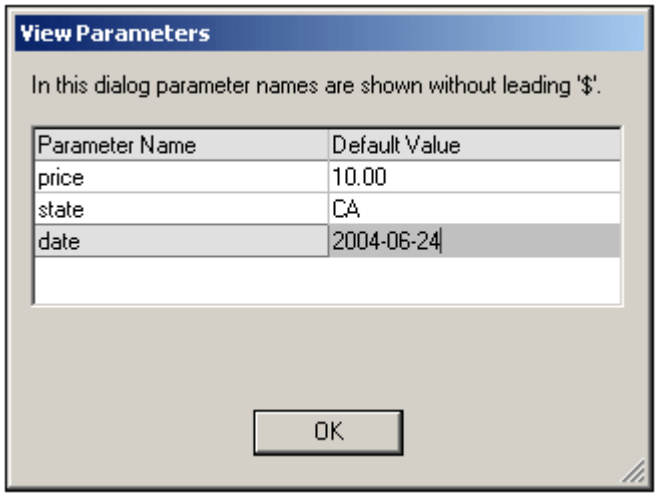
=	等しい記号
<>	等しくない記号
<	より小記号
<=	より小記号、または、等しい記号
>	より大記号
>=	より大記号、または、等しい記号
LIKE	発音が似ている
NOT LIKE	発音が似ていない
IS NULL	空である
NOT NULL	空ではない

IS NULL、または、NOT NULL が選択される場合、値フィールドは無効化されます。**値** は引用符（または、他の区切り文字）無しに入力される必要があります。値は対応するDB フィールドと同じ書式設定を持つ必要があります。それ以外の場合、式は `FALSE` を評価します。例えば、MS Access DB 内の `date` データ型のフィールドのための必要条件が式 `StartDate=25/05/2004` を持つ場

合、式は FALSE を結果として評価します。これは、MS Access DB 内の date データ型は書式 YYYY-MM-DD の書式を持つからです。

DB クエリを持つパラメーターの使用

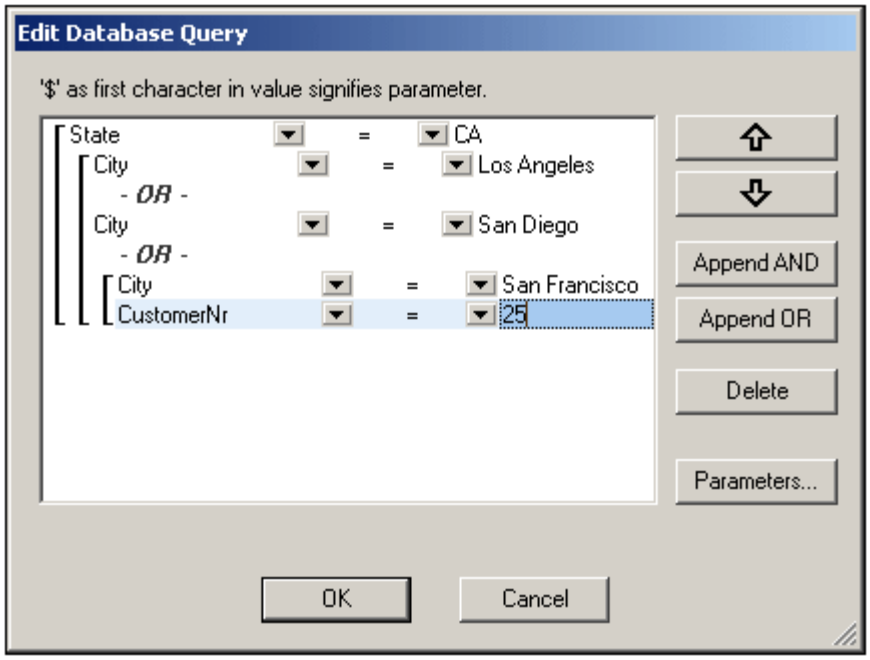
クエリを作成する再、式の値としてパラメーターの名前を入力することができます。パラメーターはクエリ内のリテラルの値の代わりに使用することのできる変数です。式内に入力すると、式内で値が使用されます。使用することのできるパラメーターは SPS 内の SPS デザイナーにより定義されており、パラメーターのビューダイアログ内で確認することができます（*下のスクリーンショットを参照してください*）。（出力ドキュメントがコマンドラインでコンパイルされる場合）コマンドラインからパラメーターに値をパスしオーバーライドすることのできる SPS 内のデフォルトの値にパラメーターが割り当てられます。SPS のために定義されたパラメーターをビューするには、データベースクエリの変更ダイアログ内の **パラメーター** ボタンをクリックします。これにより **パラメーターのビュー** ダイアログが開かれます（*スクリーンショットを参照してください*）。



SPS とパラメーター内でスタイルシートのために定義されている **すべての** パラメーターを含むパラメーターのビューダイアログはスタイルシートのデザイン内で編集される必要があります。

DB クエリ内の必要条件を再度並べ替える

DB クエリのロジカルな構造と2つの必要条件、または、必要条件のセット間のリレーションシップが視覚的に示されます。ロジカルな構造の各レベルは角かっこで示されています。2つの隣接する必要条件、または、必要条件のセットは AND 演算子を示し、2つの必要条件が OR で区切られている場合、演算子が示されます。必要条件は DB クエリのロジカルな構造のクリアな概要を与えることを目的としています。



上のスクリーンショットで表示されているDB クエリはテキストでは以下の用に表されています:

```
State=CA AND (City=Los Angeles OR City=San Diego OR (City=San Francisco AND
CustomerNr=25))
```

必要条件を移動して、または、必要条件のセットをDB クエリ内の他の条件に相対して上下に移動して、DB クエリの順番を変更することができます。必要条件、または、必要条件のセットを移動するには、以下を行います:

1. クリックして必要条件を選択し、そのレベルを示すかっこをクリックして、レベル全体を選択します。
2. ダイアログ内の「上へ」、または、「下へ」矢印ボタンをクリックします。

次の点について注意してください:

- 進行方向内の隣接する必要条件 が同じレベルの場合、2つの条件は場所を交換することができます。
- 必要条件のセット（すなわち、かっこ内の必要条件）は同じレベル内で位置を変更することができますが、レベルは変更しません。
- 同じレベル内で個別の条件は位置を変更します。隣接する必要条件が更に外向き/内向き（すなわち、同じレベルではない）の場合、選択された必要条件は外向き/内向きに1度に1つのレベル移動します。

DB クエリ内の条件を削除するには、条件を選択して「削除」をクリックします。

DB クエリの変更の方法

DB クエリの変更方法:

1. クエリボタン をクリックします。データベースクエリの変更 ダイアログボックスが開かれます。式の条件内の式を編集し、新規の条件を追加し、条件を並べ替え、DB クエリ内の条件を削除することができます。
2. 「OK」をクリックします。DB からのデータは、DB クエリへの変更が反映されるとAuthentic View に自動的に再ロードされます。

11.3.3.3 DB テーブルの変更

レコードを追加する方法

DB テーブルにレコードを追加する方法:

1. (行を追加するために)DB テーブル行 アイコン内にカーソルを置き、または、(行を入力するために) アイコンをクリックします。これにより一時的な XML ファイル内に新規のレコードが作成されます。
2. 「ファイル | 保存する」コマンドをクリックして、DB に新規のレコードを追加します。Authentic View 内では、新規のレコード内の行が DB テーブルに追加されます。このレコードのために AltovaRowStatus が A (追加済み) に設定されます。

新規のレコードのためのデータを入力すると、太字と下線と共に入力されます。これにより、既存のレコードとレコードを区別することができます。既存のレコードがテキストのフォーマット プロパティを使用して書式設定されていない場合、赤字で表示されているデータ型エラーがフラグされます。

「ファイル | 保存する」をクリックして新規のレコードは DB に追加されます。新規のレコードが DB 保存されると (— により示された) `AltovaRowStatus` フィールドが初期化され、 Authentic View 内で通常のレコードとしてレコードが表示されます。

レコードを変更する方法

レコードを変更するには、必要に応じて DB テーブル内の必要とされる箇所にカーソルを置き、レコードを編集します。表示されるレコードの数量は制限され、必要とされるレコードをナビゲートする必要がある可能性があります (次を参照してください: DB テーブルのナビゲート 440)。

レコードを変更すると、レコードの全てのフィールド内のエントリに下線が引かれ、このレコードのすべての プライマリインスタンスの `AltovaRowStatus` は U (更新済み) に設定されます。`AltovaRowStatus` を持つこのレコードのすべての セカンダリインスタンスは u (小文字) に設定されます。レコードのプライマリとセカンダリインスタンスは DB の構造と生成された XML スキーマに従い定義されます。例えば、Address テーブルが Customer テーブルに含まれる場合、Address テーブルはインスタンス化の 2 つの型内のドキュメントのデザイン内で発生することができます: これは Address テーブル自身が Customer テーブルのインスタンス化内に存在するためです。これらの 2 つの型の片方が変更されると最初に変更される型になります。他の種類としては、1 つ以上の他の型とセカンダリ型が存在する場合があります。データ型エラーは赤い色で表示されフラグされます。

「ファイル | 保存する」をクリックして変更は DB に保存されます。変更されたレコードが DB に保存されると、(— により示された) `AltovaRowStatus` フィールドが初期化され、Authentic View 内で通常のレコードとしてレコードが表示されます。

以下の点に注意してください:

- Authentic View 内のレコードの単一のフィールドが変更され、データが DB に保存されると、レコード全体が更新されます。
- 日付 値 0001-01-01 が一部の DB ために NULL 値として定義されます。エラーメッセージが発生する場合があります。

レコードを削除する方法

レコードの削除の方法:

1. 削除するレコード内にカーソルを置いて、 アイコンをクリックします。削除されるレコードは取り消し線と共にマークされます。AltovaRowStatus は以下のとおりです: レコードのプライマリインスタンスは D に設定され、セカンダリインスタンスは d に設定され、間接的に削除されるレコードは X に設定されます。間接的に削除されるレコードは個別のテーブル内に保管された削除されたレコード内のフィールドです。例えば、Address テーブルが Customer テーブルに含まれる場合があります。Customer レコードが削除される場合、対応する Address レコードが直接削除されます。Customer テーブル内の Address レコードが削除された場合、Customer テーブル名の Address レコードが主に削除されますが、インスタンス化されている場合、独立した Address テーブル内の同じレコードが 2 番目に削除されます。
2. 「ファイル | 保存する」をクリックして変更を DB に保存します。

メモ DB にデータを保存することは、元に戻すコマンドをリセットします。このため、保存前に行われたアクションを元に戻すことはできません。

11.3.4 日付と作業する

Authentic View 内で日付を編集するには2 つの方法があります:

- 日付の選択を使用して日付が入力、または、変更されます。
- 値を入力して[447] 日付が入力、または、変更されます。

Authentic View ユーザーが使用するメソッドはSPS 内で定義されています。両方のメソッドはこのセクションの2 つのサブセクションで説明されています。

日付の書式に関するメモ

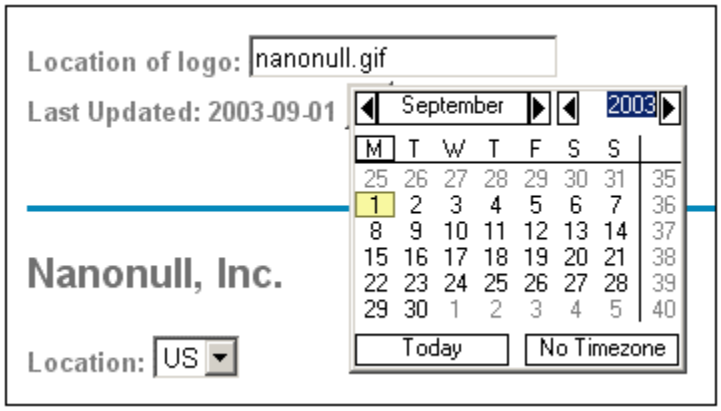
XML ドキュメント内では、日付は複数の日付データ型の1 つに保管することができます。これらのデータ型のそれぞれは、XML ドキュメントが有効であるために日付は特定の文法の書式内で保管される必要があります。例えば、`xs:date` データ型は`YYYY-MM-DD` の書式を必要とします。`xs:date` オード内の日付がこの書式以外の書式で入力されると、XML ドキュメントは無効になります。

正確な書式で日付が入力されるように、SPS デザイナーは視覚的な日付の選択をデザイン内に含むことができます。これにより、日付の選択で選択された日付が、正確な文法の書式であることを確認できます。日付の選択が存在しない場合、Authentic View は正確な文法の書式で日付の挿入を行います。XML ドキュメントの検証により必要とされる文法の書式に関するヒントを得ることができます。

11.3.4.1 日付の選択

XML ドキュメントに標準の書式で日付を挿入するために視覚的なカレンダーである日付の選択が使用されます。ドキュメント内のデータの処理のために標準の書式は重要です。日付の選択 アイコンが変更された日付フィールドの近くに表示されます（*スクリーンショットを参照してください*）。

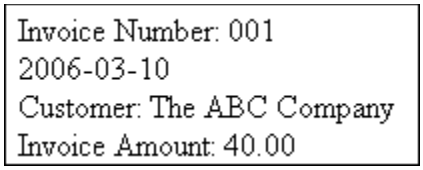
日付の選択を表示するには、日付の選択アイコンをクリックしてください（*スクリーンショットを参照してください*）。



日付を選択するには、希望する日付、付き、または、年度を選択します。XML ドキュメントに日付が挿入され、表示される日付は必要に応じて変更されます。必要な場合は、タイムゾーンも入力してください。

11.3.4.2 テキストのエントリ

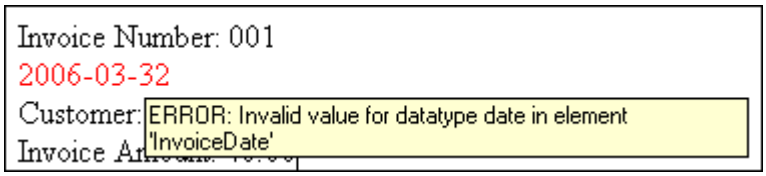
日付の選択が存在しない日付フィールドの場合、新規の値を直接入力することにより日付を編集することができます（*スクリーンショットを参照してください*）。



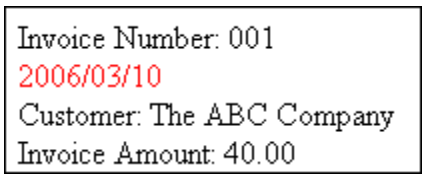
エラー

以下の種類のエラーがフラグされます:

- 日付を編集し、日付が有効な日付の範囲外になった場合、日付は赤い色で表示されエラーを通知します。マウスのカーソルを無効な日付にポイントすると、エラーメッセージが表示されます（*スクリーンショットを参照してください*）。



- 日付の書式を変更しようとすると、日付は赤い色で表示されエラーを通知します（*スクリーンショットを参照してください。ハイフォンの代わりにスラッシュが使用されています*）。



11.3.5 エンティティを定義する

エンティティに関して

ドキュメントがDTD、または、XML スキーマをベースにしているかに関わらず、Authentic View 内で使用するために エンティティを定義することができます。定義後、これらのエンティティはエンティティの入力ヘルパー内に表示され、コンテキスト メニューの「**エンティティの挿入**」サブメニュー内に表示されます。エンティティの入力ヘルパー内のエンティティをダブルクリックすると、カーソル挿入ポイントにエンティティが挿入されます。

テキスト文字列、XML フラグメント、または、他の外部リソースをドキュメント内の複数の場所で使用する場合、エンティティは役にたちます。必要とされるデータのための短い名前であるエンティティを定義するには、エンティティの定義 ダイアログを使用します。エンティティの定義後、ドキュメント内の複数の場所で使用することができます。これにより時間を節約し、効率よくメンテナンスすることができます。

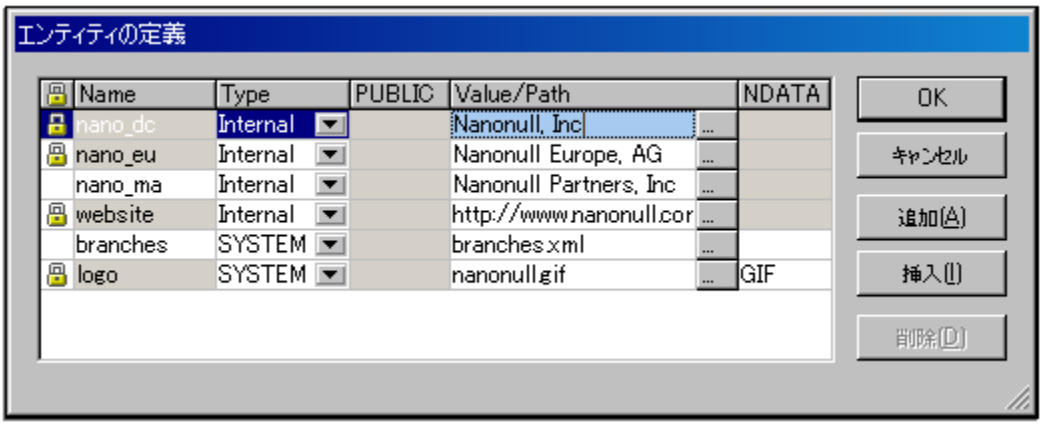
エンティティの型

ドキュメント内で使用することのできるエンティティには2つの広い型が存在します: XML データ(テキスト文字列、または、XML ドキュメントのフラグメント)である「**解析済みのエンティティ**」、または、非-XML データ(グラフィック、サウンド、または、マルチメディア オブジェクトなどの)バイナリファイルである「**解析されていないエンティティ**」。各 エンティティには名前と値が存在します。解析されたエンティティの場合、エンティティは XML データのためのプレースホルダです。エンティティの値 は XML データ 自身、または、データを含む `.xml` ファイルをポイントする URI です。解析されていないエンティティの場合、エンティティの値は非-XML データ ファイルをポイントする URI です。

エンティティの定義

エンティティを定義する方法:

1. 「Authentic | XML **エンティティの定義**....」をクリックすると、エンティティの定義 ダイアログが開かれます(*下のスクリーンショット*).



2. 名前フィールドにエンティティ名を入力します。これは、エンティティの入力ヘルパーに表示される名前です。
3. エンティティの型を型フィールドのドロップダウンリストから入力します。次の型を使用することができます: テキストが使用する **Internal** エンティティが XML ドキュメント自身内に保管されています。**PUBLIC** または **SYSTEM** の選択はリソースが XML ファイルの外部にあることを指定し、公開識別子または システム識別子を使用してロケートされます。システム識別子はリソースの場所を与える URI です。公開識別子はプロセッサーがリソースを識別するためのロケーションに影響されない識別子です。公開とシステム識別子の両方を指定すると、公開識別子はシステム識別子、システム識別子を使用して解決します。
4. PUBLIC が Type として選択されている場合、PUBLIC フィールド内にリソースの公開識別子を入力します。**Internal**、または、型としてシステムを選択すると、PUBLIC フィールドは無効化されます。
5. Value/Path フィールド内に、次を入力することができます:

- エンティティの型がInternal の場合、エンティティの値として使用するテキスト文字列を入力します。エントリに引用符を入力しないでください。入力する引用符はテキスト文字列の一部として扱われます。
- SYSTEM がエンティティの型である場合、参照ボタンを使用してリソースのURI 、または、ローカルネットワーク上のリソースを入力します。解析されたデータがリソースに含まれている場合、XML ファイルである必要があります（すなわち、ファイルは.xml 拡張子を持つ必要があります）。更に、リソースはGIF ファイルなどのバイナリファイルであることができます。
- エンティティの型がPUBLIC の場合、このフィールドにシステム識別子を追加で入力する必要があります。

6. NDATA エントリはプロセッサーにこのエンティティを解析せず適切なプロセッサーに送信されるように命令します。NDATA フィールドはエンティティが解析されていないエンティティを示す値を含む必要があります。

ダイアログの機能

エンティティの定義 ダイアログ内で以下の機能を使用することができます:

- エンティティの追加
- エンティティの挿入
- エンティティの削除
- 列ヘッダーをクリックしてエンティティをアルファベット順の値に並べ替えます。1度クリックすると昇順に並べ替え、2度クリックすると降順に並べ替えられます。
- ダイアログボックスと列の幅のサイズを調整します。
- ロック。XML ドキュメント内でエンティティが使用されると、このエンティティはロックされ、エンティティの定義 ダイアログ内で編集することはできません。ロックされたエンティティは、最初の列内でロックシンボルにより表示されます。エンティティをロックすることにより、エンティティに対してXML ドキュメントが有効であることを保証することができます（エンティティが参照されており、定義されていない場合は、ドキュメントは無効になります）。
- エンティティの複製はフラッグされます。

エンティティの制限

- 他のエンティティ内に含まれるエンティティはダイアログ Authentic View、またはXSLT 出力内で解決されません。このようなエンティティのアンパサンド文字は& などのエスケープ文字と共に表示されます。
- イメージファイルではない外部の解析されていないエンティティはAuthentic View 内で解決されません。デザイン内のイメージは外部の解析されていないエンティティを読み取るように定義され、URI がエンティティ名（例: `'logo'`）になるように設定されています。エンティティの定義 ダイアログ内でエンティティ名を（上のスクリーンショット内のロゴエンティティのために設定されているように）イメージファイルのURI に対して解決する値を持つ外部の解析されていないエンティティとして定義することができます（*上のスクリーンショットを参照してください*）。

11.3.6 XML 署名

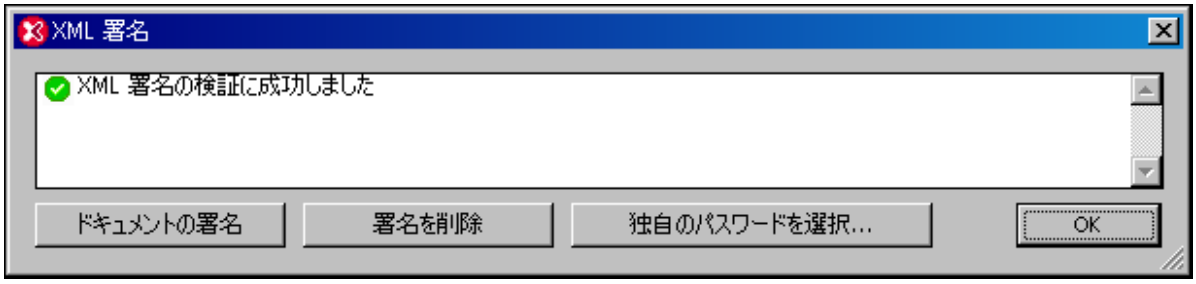
Authentic View のために構成されたXML 署名を使用してSPS をデザインすることができます。XML 署名がSPS 内で有効化された場合、Authentic View ユーザーは有効化された署名を使用してAuthentic XML ファイルを電子的に署名することができます。ドキュメントの署名後、変更を加えると署名の検証は失敗します。署名されたAuthentic XML ドキュメントがAltova 製品内のAuthentic View で開かれると、ドキュメントの検証プロセスが実行され、検証の結果はウィンドウ内に表示されます。

メモ 次のAltova 製品のEnterprise とProfessional のAuthentic View 内でXML 署名を使用し、検証することができます: Authentic Desktop、Authentic Browser、XMLSpy、とStyleVision。

XML 署名アクション

次の署名のためのAuthentic View ユーザーアクションを使用することができます:

- *証明書/パスワードの選択:* 署名は、証明書、または、パスワードで認証されます。署名が作成され、検証される際に認証オブジェクト（証明書、または、パスワード）が必要とされます。Authentic XML ドキュメントが署名が有効化されている SPS を割り当てられているかにより、デフォルトの証明書、または、署名のパスワードを指定する場合があります。デフォルトの証明書、または、パスワードが指定されているかにより、Authentic View ユーザーが自身の証明書とパスワードを選択できるかを決定する署名を構成することができます。XML 署名ダイアログ内でこの操作を行うことができます（*下のスクリーンショット*）。自身の証明書及びパスワードの選択はデフォルトの証明書とパスワードをオーバーライドします。自身の証明書とパスワードはメモリ内に保管され、現在のセッションで使用することができます。自身の証明書とパスワードの選択の後、ファイルまたはアプリケーションを閉じます。SPS は証明書とパスワードのためのデフォルトの設定に戻ります。



- *ドキュメントの署名:* Authentic XML ドキュメントを自動的に、または、手動で署名することができます。自動署名は、SPS デザイナーにより署名の構成内で指定することができ、保存されると Authentic XML ドキュメントが自動的に署名されます。自動署名が有効化されていない場合、ドキュメントを手動で署名することができます。XML 署名 ツールバーアイコン）すべての、または、「**Authentic | XML 署名** コマンドをクリックすると、XML 署名 ダイアログがポップアップされるので、（*上のスクリーンショット*）**ドキュメントの署名** ボタンをクリックします。埋め込まれた署名を使用したドキュメントの署名はスキーマのルート（ドキュメント）要素の最後の子要素として署名要素を許可するように要求します。ドキュメントを署名する場合、認証オブジェクトと署名の配置は署名の構成に従い決定されます。認証情報にアクセスできることを確認してください。詳細に関しては、SPS デザイナーに確認してください。

- *Authentic XML ドキュメントの検証:* SPS のために XML 署名が有効化されていると、検証プロセスは Authentic View XML ドキュメントがロードされる都度、各署名に対して行われます。パスワード、または、証明書 キー情報が SPS と署名と保存されていない場合、Authentic View ユーザーはパスワードを入力、および、検証のための証明書を選択するようにプロンプトされます。XML ファイルと共に保存しない場合、埋め込まれた署名が生成されると、XML ファイルが保存される際に XML ファイルと共に保存されます。（XML 署名ダイアログの「**署名を削除**」ボタンを使用して）生成された署名は明示的に削除される必要があります 同様に、デタッチされた署名が生成されると、必要では内場合、明示的に削除されます。

11.3.7 Authentic View 内のイメージ

Authentic View を使用すると、最終出力ドキュメント（HTML、RTF、PDF および Word 2007）内で使用することのできるイメージを指定することができます。イメージの書式の一部は一部の書式、または、アプリケーションでサポートされない場合があることに注意してください。例えば、SVG 書式 は PDF 内でサポートされていますが、RTF ではサポートされておらず、HTML 内で表示するためにはブラウザーアドオンが必要になります。イメージ 書式を選択する場合、ドキュメントの出力書式でサポートされている書式を選択してください。
イメージの書式の多数は、多くの出力書式でサポートされています（*下のリストを参照してください*）。

Authentic View は Internet Explorer をベースにしており、Internet Explorer が表示することのできるイメージの書式の大部分を表示することができます。次の一般的なイメージの書式がサポートされています:

- GIF
- JPG
- PNG
- BMP

- WMF (Microsoft ウィンドウ メタファイル)
- EMF (拡張されたメタファイル)
- SVG (PDF 出力のためのみ)

相対的なパス

相対的なパスはSPS ファイルに相対して解決されます。

11.3.8 Authentic View 内のキーストロール

Enter キー

Authentic View 内では、「Enter」キーは、特定のカーソルの場所で追加の要素を追加する際に使用されます。例えば、(スキーマに従い)本のチャプターには複数の段落が含まれており、段落のテキスト内で「Enter」を押すと、新規の段落が現在の段落の後に追加されます。チャプターがタイトルと複数の段落を含むことができる場合(タイトル要素を含む)、段落要素外のチャプター内で「Enter」を押すと、(スキーマにより複数のチャプターが許可されていると想定し)現在のチャプターの後に新規のチャプターが追加されます。

メモ 「Enter」キーは新規のラインを入力しません。これは、段落などのテキストノード内にカーソルがある場合でも同じです。

キーボードの使用

入力とナビゲートのためにキーボードを通常の方法で使用することができます:

- 「タブ」キーはカーソルを前に進め、ノードの前後で停止し、ノードコンテンツをハイライトし、静的なコンテンツをステップオーバーし、静的なコンテンツをステップオーバーします。
- `add...` と`add Node` ハイパーリンクは、ノードコンテンツとして考えられ、タブ付けされるとハイライトされます。スペースキーまたは、「Enter」キーを押すと有効化することができます。

11.4 Authentic スクリプト

Authentic スクリプト機能により、SPS デザインに対して更なる柔軟性と双方向性を追加することができます。これらデザインは StyleVision Enterprise ならびにProfessional Edition で作成または編集することができ、Altova 製品のEnterprise ならびに Professional Edition に搭載されたAuthentic ビューにて閲覧することができます。

以下のテーブルではAltova 製品ごとにサポートされる機能のリストを示します。Authentic Browser Plugin Trusted バージョンはセキュリティー上の観点から内部スクリプトが無効になっていることにご注意してください。

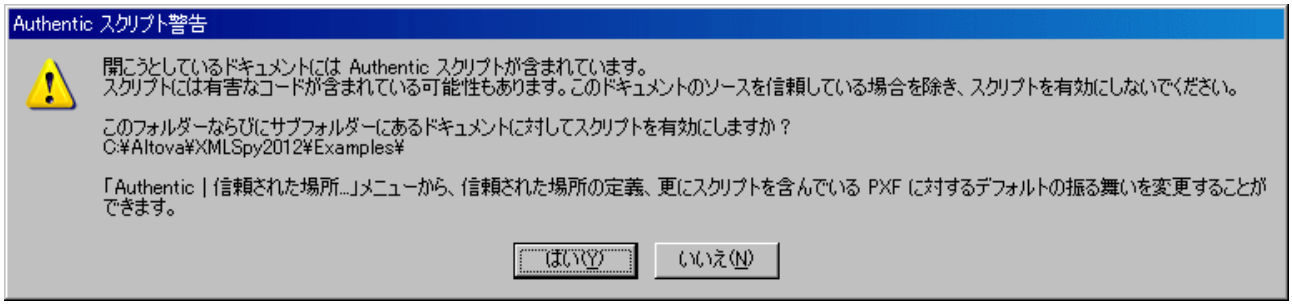
Altova 製品	Authentic スクリプトの作成	Authentic スクリプトの有効化
StyleVision Enterprise	はい	はい
StyleVision Professional	はい	はい
StyleVision Standard *	いいえ	いいえ
XMLSpy Enterprise	いいえ	はい
XMLSpy Professional	いいえ	はい
XMLSpy Standard	いいえ	いいえ
AuthenticDesktop Enterprise	いいえ	はい
AuthenticDesktop Community	いいえ	いいえ
Authentic Browser Plug-in Community **	いいえ	いいえ
Authentic Browser Plug-in Enterprise Trusted ***	いいえ	はい
Authentic Browser Plug-in Enterprise Untrusted	いいえ	はい

* *Authentic ビュー無し*
** *Trusted ならびにUntrusted バージョン*
*** *スクリプトありのデザインは表示可能。内部マクロの実行やイベントハンドリングは無し。外部イベントの実行。*

Authentic スクリプトの動作は全 Altova 製品において同一のものとなり、ある製品に特化したコードや設定は必要ありません。

Authentic スクリプト警告ダイアログ

PXF ファイルまたはSPS へリンクされたXML ファイルにスクリプトが含まれており、そのファイルが開かれた状態でAuthentic ビューへの切り替えが行われると、警告ダイアログが表示されます(以下のスクリーンショットを参照)。



以下のオプションから選択を行うことができます:

- **はい**をクリックすると、ファイルを含んでいるフォルダーがAuthentic スクリプトの信頼された場所リストへ追加されます。今後、信頼されたフォルダー内にふくまれるファイルをAuthentic ビューで開いても、警告ダイアログは表示されなくなります。信頼された場所のリストへは「Authentic | 信頼された場所 910」メニューコマンドからアクセスならびに編集することができます。
- **いいえ**をクリックすることで、ファイルが含まれているフォルダーを信頼された場所のリストへ追加することなく処理が継続されます。ファイルはスクリプトが無効になった状態で開かれます。ファイルをAuthentic ビューで開くたびにAuthentic スクリプト警告ウィンドウが表示されます。ファイルが含まれているフォルダーを信頼された場所のリストへ追加するには、「Authentic | 信頼された場所 910」メニューコマンドから信頼された場所のダイアログを開き、フォルダーを追加ならびに必要に応じて変更することができます。

信頼された場所ダイアログに関する詳細については、ユーザーリファレンスの「Authentic | 信頼された場所 910」メニューコマンドのセクションを参照ください。

メモ: COM インターフェースを介したXMLSpy へのアクセスを行う場合、**セキュリティーのチェックは行われず**、「**Authentic スクリプト警告**」**ダイアログは表示されません**(COM インターフェースに関する情報はプログラマーのレファレンス 1054 を参照ください)。

Authentic スクリプトの使用方法

SPS デザインを作成したデザイナーは以下にある2つの方法により、Authentic ドキュメントをインタラクティブなものにすることができます:

- スクリプトをユーザー定義のアクション(マクロ)に割り当てることで、要素、ツールバーボタン、そしてコンテキストメニューのアイテムをデザイン
- Authentic ビューイベントに反応するデザインイベントハンドラーを追加

Authentic ドキュメントをインタラクティブなものにするために必要な全てのスクリプトは、StyleVision GUI (Enterprise ならびに Professional Edition)内で作成することができます。フォーム、マクロ、イベントハンドラーはStyleVision に搭載されたスクリプトエディター内で作成され、これらスクリプトはSPS 内に保存されます。その後、StyleVision のデザインビューにて、保存されたスクリプトは要素、ツールバーボタン、コンテキストメニューに割り当てられることになります。SPS をベースにしたXML ドキュメントがAuthentic スクリプトをサポートするAltova 製品(上のテーブルを参照)により開かれると、ドキュメントには新たな柔軟性や双方向性が追加されます。

Authentic スクリプトのドキュメンテーション

Authentic スクリプトに関する情報はStyleVision のドキュメンテーションでご利用になれます。Altova ウェブサイトの製品ドキュメンテーションページから参照することができます。

12 HTML と CSS

XMLSpy では、HTML 455、とCSS 457 ドキュメントに対するインテリジェントな編集機能が提供されます。全てのドキュメントはテキストビュー 122 にて編集することができ、アクティブな HTML はブラウザービューにてプレビューを行うことができます。

各ドキュメントの種類に関するインテリジェント編集機能についてはこのセクション以下のサブセクションに記されます：HTML 455、と CSS 457。

(C) 2015-2021 Altova GmbH

12.1 HTML

テキストビューではHTMLドキュメントを編集することができ、編集された内容を、すぐにブラウザービューで確認することができます。テキストビューでは以下にあるような数々のHTML 編集機能が提供されています。詳細はテキストビュー122のセクションを参照ください。

サポートレベル

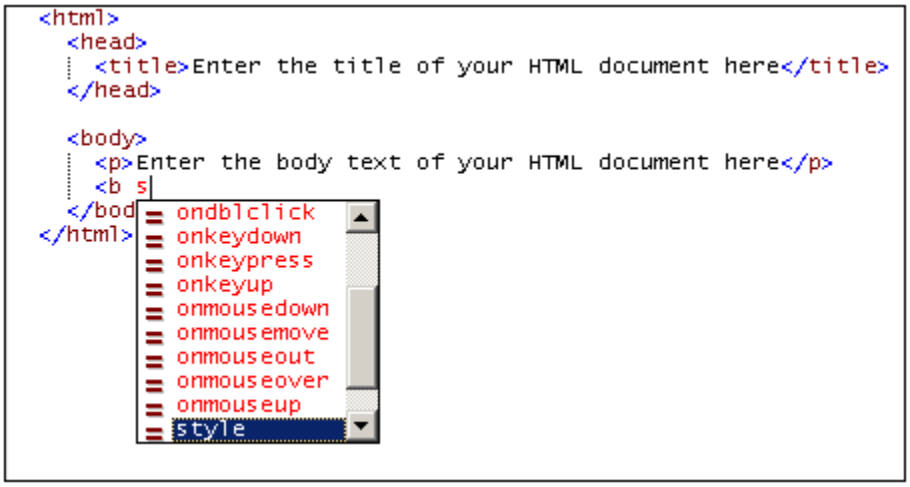
XMLSpy ではHTML 4.0 ならびにHTML 5.0 がサポートされます。入力ヘルパーならびにインテリジェントな編集機能が対応したHTML のバージョンに対して利用可能です。これらの機能を以下に記します。

入力ヘルパー

HTMLドキュメントがアクティブになっていれば、要素、属性、エンティティ入力ヘルペを利用することができます。入力ヘルパーの内容は文脈により変化し、現在カーソルのある位置に入力することができるアイテムが入力ヘルパーには表示されます。HTML 入力ヘルパーはテキストビュー138にあるその他の入力ヘルパーと同じように使用することができます。

自動補完

HTMLドキュメントにマークアップを入力する際、XMLSpy は自動補完ヘルプを表示します。ポップアップにはカーソルのある位置に挿入することができるノード全てが表示され、入力がなされるごとに、リスト内にある最も近いマッチの先頭に選択が移動します（スクリーンショットを参照）。選択されたアイテムをクリックすることで、そのアイテムがカーソル位置に挿入されます。



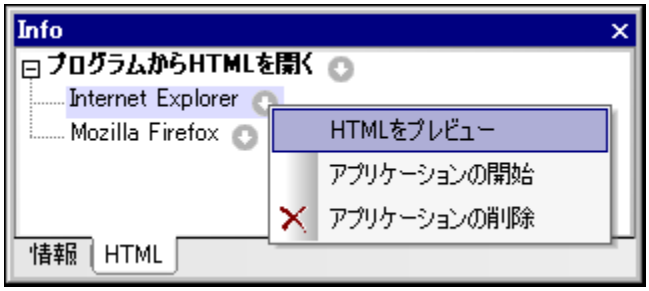
要素に対する自動補完は、ノード開始タグが入力されたときに表示されます。ノードの開始タグがドキュメントに入力されると、その終了タグも自動的に挿入されます。この機能により整形式が保証されます。

ノードの開始タグにて要素名のあとにスペースが入力されると、属性に対する自動補完が表示されます。自動補完ポップアップにある属性名をクリックすると、属性名と引用句が入力され、カーソルが引用句の中間に移動します。

エンティティ入力ヘルパーにはHTML 4.0 、とHTML 5.0エンティティセット Latin-1、特殊な文字、そしてシンボルが含まれています。

HTML 情報ウィンドウ

（以下のスクリーンショットにある）HTML 情報ウィンドウを使えば、アクティブな HTML ファイルに素早くアクセスすることができます。例えば、HTML ファイルがXMLSpy でアクティブであれば、HTML 情報ウィンドウ内にある Mozilla Firefox アイテムをダブルクリックすることで、Mozilla Firefox が起動され、そのHTMLドキュメントがロードされます。



以下の点に注意してください:

- *プログラムから HTML を開く* アイテムの右にあるアイコンを使うことで、*プログラムから HTML を開く* リストにアプリケーションを追加することができます。システムにインストールされているブラウザーまたはその他の(テキストエディターのような)アプリケーションを追加することができます。通常追加されるのはブラウザーまたはエディターアプリケーションです。
- アプリケーションが *プログラムから HTML を開く* リストに追加されたら、(「**インストールされたブラウザーを追加**」コマンドを使用した場合を除いて)アイテムを選択して、「**F2**」を押すことで、リスト内に表示されるアプリケーションの名前を変更することができます。
- リストの各アイテムの右側にあるアイコンを使えば、(i) アプリケーションを開始し、(ii) アプリケーションを開始した後でリンクされた HTML ファイルを開くか、(iii) リストからアプリケーションを削除することができます。アプリケーション名をダブルクリックすることで、リンクされた そのアプリケーションで HTML ファイルが開かれます。
- プログラムから HTML を開く リストに表示されるアプリケーションは、CSS 情報ウィンドウのアプリケーションアイテムと連動しています。

DTD の割り当て

「**DTD/スキーマ**」メニューを使えば、XHTML ドキュメントに対して DTD または XML スキーマを参照、割り当てることができます。XHTML ドキュメントの編集はXML ドキュメントと同じように 234行うことができます。

ブラウザービューコマンド

ブラウザービューコマンドは「**ブラウザー**」メニューから使用することができます。

12.2 CSS

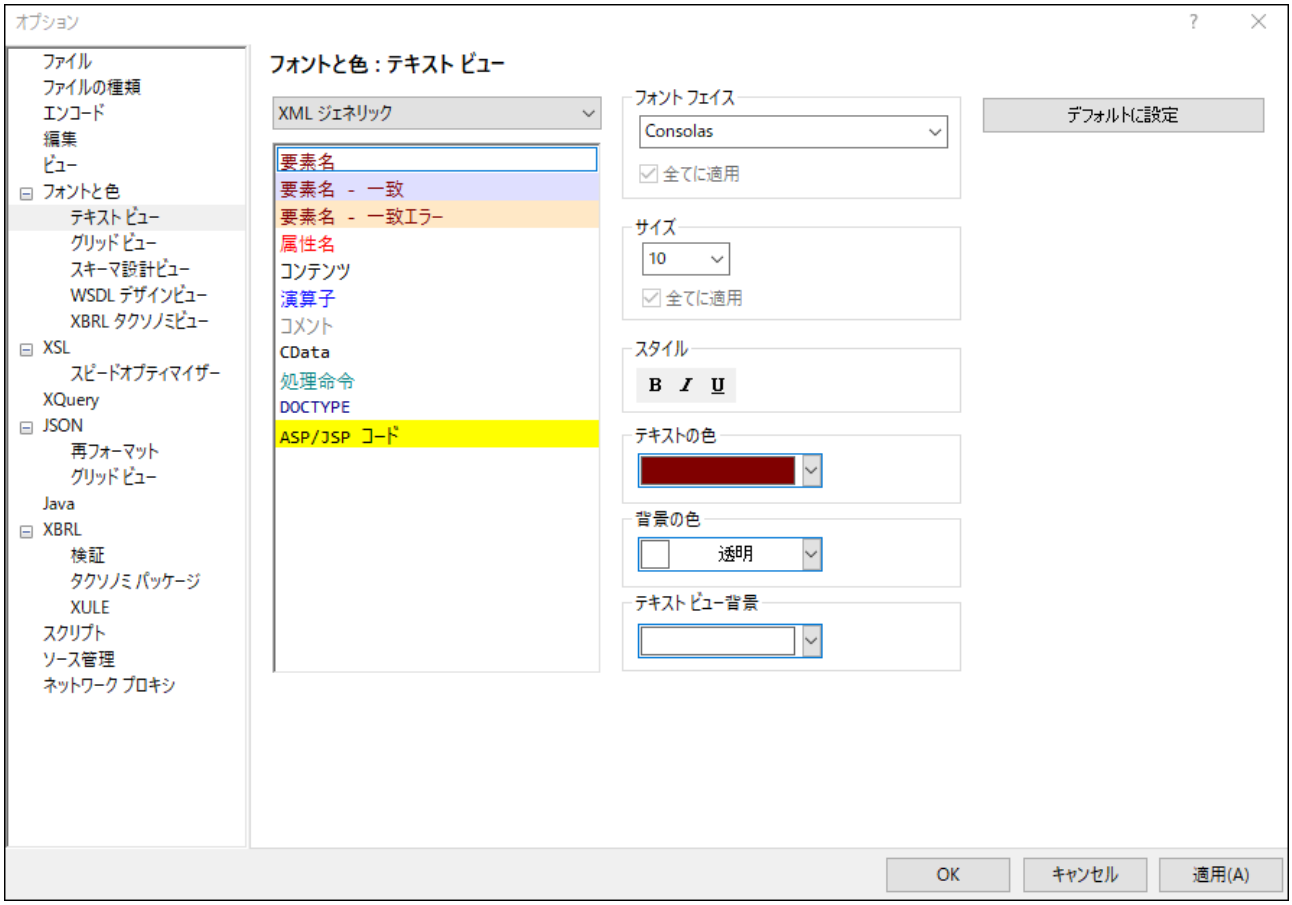
テキストビューのインテリジェント編集機能を使用して CSS ドキュメントの編集を行うことができます。CSS の編集に利用することができる編集機能は以下のとおりです。

構文色分け

CSS ルールはセレクター、(1つ以上の) プロパティ、そしてそれらプロパティの値により構成されます。これら3つのコンポーネントは、細かな指定を行うため、更に細かなカテゴリに分割されることもあります。例えば、セレクターにより指定されるものにクラス、擬似クラス、ID、要素、または属性があります。更に、CSS ドキュメントは(例えばコメントのような) ルール以外のアイテムも含むことができます。テキストビュー内ではこのようなアイテムのカテゴリはオプションダイアログ内で設定された設定に従い異なる色で表示されます (*下のスクリーンショット*) (*下のスクリーンショット参照*)。

```
.header
{
    font-family: "Arial", sans-serif;
    font-weight: bold;
    color: red;
}
```

オプションダイアログのフォントと色のセクションで多種の CSS コンポーネントの色を設定することができます (*下のスクリーンショット*). (以下のスクリーンショットにあるように) オプションダイアログのテキストフォントタブから、様々な CSS コンポーネントの色を指定することができます。



ソースの折りたたみ

ソースの折りたたみは、ソースの折りたたみ余白内で +/- サインにより表示される、各 CSS ルールを展開または折りたたむ機能を指します。余白はテキストビュー設定ダイアログ969 により切り替えのオンとオフを行うことができます。ルールが折りたたまれると、視覚的に省略記号により表示されます。マウスのカーソルが省略記号にポイントされると、折りたたまれたルールの内容がポップアップ内に表示されます。内容がポップアップに大きすぎる場合、これは、ポップアップの下に省略記号で表示されます。

テキストツールバー内の **全ての折りたたみの切り替え** アイコン はすべての ルールを切り替え、トップレベルのドキュメント要素に対するルールを展開または折りたたみます。

メモ 中かっこのペアは、ルールを区切り、(*上のスクリーンショット*) 中かっこの前または後ろにカーソルがポイントされると太字で表示されます。これにより特定のルールがどの位置で開始し終了するかの定義が表示されます。

CSS アウトライン

(*以下のスクリーンショット*) CSS アウトライン入力ヘルパーでは、ドキュメント内にあるセレクターが表示されます。CSS アウトラインにあるセレクターをクリックすると、ドキュメント内の対応する箇所がハイライトされます。左下にあるスクリーンショットでは、セレクターの並べ替えは行われず、ドキュメント内で表示される順番通りに表示されています。右下のスクリーンショットでは、(ツールバーアイコンの) アルファベット順の並び替え機能が有効になっており、セレクターがアルファベット順に並べ替えられています。

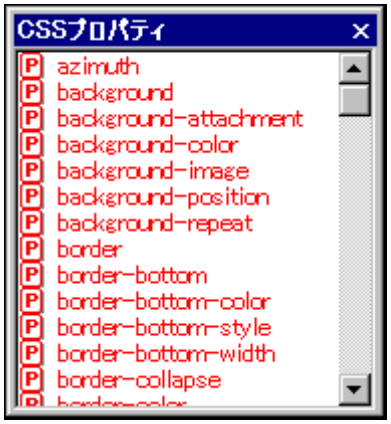
以下の点に注意してください:(i) セレクターの並べ替えを行う際には、(ピリオド、ハッシュ、コロンといった) セレクター全ての箇所が並べ替え評価の対象になり、(ii) 複数のセレクター(例:`h4, h5, h6 {...}`) が1つのルールとしてグループ化されている場合でも、CSS アウトラインではグループ内の各セレクターは別々に表示されます。

CSS アウトラインイン入力ヘルパーにあるツールバーアイコンを左から順に説明します:

	ドキュメントとの自動同期の有効/無効を切り替えます。自動同期が有効になっている場合、ドキュメント内で入力されたセレクターが(入力中でも)入力ヘルパーに表示されるようになります。
	入力ヘルパーの内容を現在のドキュメントの状態から更新します
	アルファベット順による並び替えの有効/無効を切り替えます。オフになっている場合、セレクターはドキュメント内に記述されている順序で表示されます。アルファベット順で並び替えられた場合、(例: `#intro` などの)ハッシュ記号により記述されるID が最初のセレクターとして表示されます。

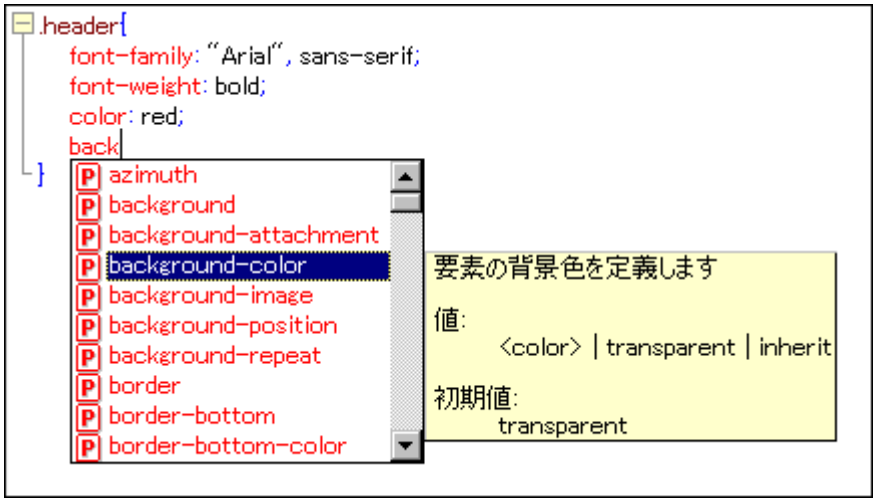
プロパティ入力ヘルパー

(*以下のスクリーンショットにある*)プロパティ入力ヘルパーには、全てのCSS プロパティがアルファベット順に表示されます。プロパティのアイテムをダブルクリックすることで、カーソルのある位置にそのプロパティが挿入されます。



プロパティの自動補完とプロパティのヒント

プロパティの名前を入力しようとすると、XMLSpy により、入力された文字から始まるプロパティのリストが表示されます(*下のスクリーンショット*)。その他にも、カーソルをプロパティ名の中まで移動させ、**Ctrl+Space** を押下することでCSS プロパティのリストが表示されます。

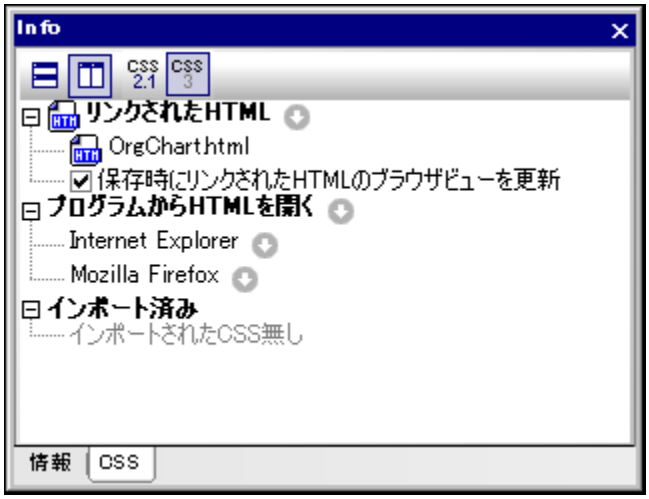


リストをスクロール、または矢印キーによりリストのナビゲーションを行うことで、プロパティの定義と使用可能な値が記されたツールチップが表示されます。表示されるツールチップはハイライトされたプロパティのものです。選択されたプロパティを挿入するには、「Enter」を押下するか、プロパティをクリックしてください。

CSS 情報ウィンドウ

CSS ファイルがアクティブになっていれば、CSS 情報ウィンドウが有効になります(*下のスクリーンショット*)CSS 情報ウィンドウでは以下の機能が利用可能です:

- CSS 2.1 とCSS 3.0 との切り替えを行うことができます。GUI における入力ヘルパーとインテリジェントな編集機能が、情報ウィンドウで選択されているCSS のバージョンにより切り替わります。
- CSS ファイルを HTML ファイルにリンクすることができます。この機能によりCSS ドキュメントの修正により反映される結果をただちに確認することができます。他にも、リンクされた HTML を複数のブラウザーで表示することができ、CSS ドキュメントの修正を異なるブラウザーにて確認することができます。
- CSS 情報ウィンドウにはインポートされたCSS スタイルシートが表示され、アクティブな CSS ドキュメントにインポートされたファイル構造の概要を知ることができます。



以下の点に注意してください:

- 情報ウィンドウのツールバーにはCSS 2.1 とCSS 3.0 のアイコンが含まれます。目的のバージョンを選択することで、選択したバージョンのCSS に対応した入力ヘルパーならびに編集機能を使用することができます。
- アクティブな CSS ドキュメントからリンクすることのできるHTML ファイルは1つだけです。*リンクされた HTML* の左にあるアイコンをクリック、「HTML リンクを設定」コマンドを選択し、目的のHTML ファイルを選択します。選択されたファイルが*リンクされた HTML* アイテムの下に表示されます。このリンクを作成してもCSS やHTML ドキュメントへの修正が施されるわけではありません。このリンク機能により、アクティブな CSS ファイルを適用して、ファイルを試験することができます。
- リンクされたHTML ファイルをダブルクリックすることで、そのHTML ファイルがXMLSpy により開かれます。
- ツールバーアイコンにより、CSS ファイルとHTML ファイルを水平、または垂直に配置することができます。
- 変更されたCSS ドキュメントを保存する際、XMLSpy で開かれているHTML ファイルを自動的に更新することができます。この機能を有効にするには、*保存時にリンクされた HTML のブラウザービューを更新* ボックスにチェックを入れてください。この機能による自動更新は、編集されたCSS ドキュメントへの参照を行っているHTML ファイルに対してしか行われないことに注意してください。
- リンクされたHTML ファイルを変更するには、「HTML リンクを設定」コマンドから、別のHTML ファイルを選択してください。
- HTML ファイルへのリンクを削除するには、*リンクされた HTML ファイル* の右にあるアイコンをクリックし、「リンクを削除」コマンドを選択してください。
- プログラムからHTML を開く の右にあるアイコンを使うことで、リストへアプリケーションを追加することができます。コンピューターにインストールされたブラウザーまたは(テキストエディターの様な)任意のアプリケーションをメニューコマンドに追加することができます。割り当てを行うアプリケーションは、通常ブラウザーかエディターです。
- アプリケーションがプログラムが *HTML を開く* リストに追加された後は、(「インストールされたブラウザーを追加」コマンドにより追加されたものを除いて) *HTML を開く* リスト内の名前は選択範囲を選択し、「F2」キーを押下することにより名前の変更を行うことができます。
- *HTML を開く* に表示されているアプリケーションの右にあるアイコンからは、(i) アプリケーションを開く、(ii) アプリケーションを開き、リンクされているHTML ファイルを開く、(iii) リストからアプリケーションを削除するコマンドを選択することができます。アプリケーション名をダブルクリックすることで、リンクされているHTML ファイルをそのアプリケーションで開くことができます。
- *HTML を開く* リストに(から)追加/削除されたアプリケーションは、HTML 情報ウィンドウのアプリケーションとも連動しています。
- *インポート済み* アイテムには、アクティブな CSS ファイルにインポートされたCSS ファイルが表示されます。

13 JSON と JSON スキーマ

JSON（ジェイソン、JavaScript Object Notation）は、軽量のデータストレージとJavaScript 構文を使用する交換形式です。また、XML のように人間が判読できる、テキストのみの書式です。JSON テキストは、すべてプログラミング言語で読み込まれ、使用されることができるので、広範囲におよび、特にウェブ上で、データ交換書式として使用されるようになりました。

IDE 機能の一部として、XMLSpy は、JSON データ（インスタンスドキュメント）[464] の編集および検証をサポートし、構文的にも、文法的にも正確な JSON スキーマ[467] ドキュメントの作成をサポートします。

JSON5

ECMAScript 5 拡張を追加するJSON の拡張がJSON5 です（詳細に関してはjson5.org を参照してください）。JSON5 は、JavaScript の厳密なサブセットあり、新規のJSON データ型を追加せず、全ての既存のJSON コンテンツと作業することができます。

JSON インスタンスドキュメントのために使用することのできる、すべてのXMLSpy 機能は、JSON5 インスタンスドキュメントに対して使用することができます。しかしながら、JSON5 とJSON の間には以下の主要な違い、および、これら2 種類のフォーマットをXMLSpy が扱う方法の違いが存在します:

- JSON5 はJSON の公式な継承ではありません。このため、自身のファイル拡張子を使用しています: `json5`。
- デフォルトでは、XMLSpy は、`.json` ファイル拡張子をJSON インスタンスドキュメントとして認識します。`.json5` ファイル拡張子を持つインスタンスドキュメントをJSON5 インスタンスドキュメントとして認識します。
- JSON5 インスタンスドキュメントはJSON スキーマに対して検証することができます。

このドキュメント内では、*JSON インスタンスという用語は*、指定されない限り、JSON とJSON5 インスタンスドキュメントを指します。次を参照してください: JSON5 とJSON 間の違い[465]。

XMLSpy 内のJSON、JSON スキーマ

JSON インスタンスとJSON スキーマ の両方のドキュメントの型は、JSON フォーマットで書かれており、有効性のために整形式のJSON ルールに準拠しなければなりません。（インスタンスとスキーマ）の両方のドキュメントは、通常、`.json` ファイル拡張子を持っています。JSON インスタンスは、テキストビュー[470] とグリッドビュー[475] で編集することができ、JSON スキーマドキュメントをこれら2 つのビュー、および、視覚的なスキーマビューであるJSON スキーマビュー[505] で、編集することができます。

XMLSpy は、JSON インスタンスとJSON スキーマドキュメントとの作業のために以下をサポートしています:

- テキストビュー[470] には、構文の色分け、構文のチェック、JSON スキーマの関連性がある場合のインスタンスドキュメント内の自動補完、余白の折り返し、構造マーキングなどの機能を使用して、有効な JSON インスタンスとJSON スキーマドキュメントの編集を素早く行うことができます。テキストビュー[470] には、インスタンスおよびスキーマドキュメントの検証が搭載されています。
- グリッドビュー[475] には、ドキュメント構造を更にビジュアル化するテーブルグリッド構造が搭載されています。グリッドビュー[475] で直接編集することができます。切り替え可能な テキストビュー[470] および グリッドビュー[475] は、編集に関するニーズに応えます。グリッドビュー[475] には、インスタンス および スキーマドキュメントの検証が搭載されています。
- テキストビュー[470] および グリッドビュー[475] 内のJSON インスタンスドキュメント検証。情報ウィンドウ[537] 内で割り当てられたJSON スキーマに対して検証は実行されます。
- JSON スキーマビュー[505] は、JSON スキーマをグラフィカルなレイアウトで表示します。これにより、JSON スキーマを素早く作成するためのドラッグアンドドロップ機能を（テキスト入力に加え）使用することができます。ビュー内の入力ヘルパーは、編集入力を与えます。更に、スキーマの有効性が継続的にチェックされ、エラーが発生するとフラッグされます。

JSON インスタンス: 既存のインスタンスドキュメントを開き、新しいインスタンスドキュメントを作成する

- 「オプション | ファイルの型」[1018] タブ内で、JSON/JSON5 インスタンスドキュメントを開くためのデフォルトのビュー（テキストビュー[470] または グリッドビュー）[475] を設定することができます。既存のJSON/JSON5 ドキュメントは、選択されたデフォルトのビューで開かれます。テキストビュー[470] とグリッドビュー[475] をいつでも切り替えることができます。

(C) 2015-2021 Altova GmbH

- 新しいJSON またはJSON5 インスタンスドキュメントを作成するためには、「**ファイル | 新規作成**」をクリックして、`json: JavaScript Object Notation` または`json5: JSON with ECMAScript 5 extensions.` をそれぞれ選択します。任意で、新しいインスタンスファイルのためのJSON スキーマファイルを選択するように問われます。スキーマを割り当てるには、割り当ては情報ウィンドウ[537]内で入力されます。「オプション | ファイルの型」[1018] タブ内の設定に従い、新規 インスタンスドキュメントは、テキストビュー[470] またはグリッドビュー[475]で開かれます。

JSON スキーマ: 既存のスキーマを開く、または、新しいスキーマを作成する

- 既存のJSON スキーマドキュメントをJSON スキーマビュー[505]で開きます。テキストビュー[470] またはグリッドビュー[475] にいつでも切り替えることができます。
- 新規 JSON スキーマドキュメントを作成するには、「**ファイル | 新規作成**」をクリックして、`j son: JSON スキーマ` を選択します。ドキュメントの最初に`$schema` キーワードを持つ、新規 JSON スキーマドキュメントが JSON スキーマビュー[505]で開かれます。テキストビュー[470] またはグリッドビュー[475] にいつでも切り替えることができます。

これら全てのビュー（テキストビュー[470] グリッドビュー[475] およびJSON スキーマ[505]）に関しては、このセクションのサブセクションで説明されています。

このセクション

このセクションは以下のトピックに整理されています:

- JSON データについて[464] は、JSON ドキュメントの基本的情報について説明しています。
- JSON スキーマについて[467] は、JSON スキーマについてまたその作動について説明しています。
- JSON ラインとJSON コメント[469] ではXMLSpy によりサポートされる2 つのJSON 追加仕様についての情報が提供されています。
- テキストビュー内のJSON ドキュメント[470] は、テキストビューのJSON に関連した機能と作業する方法について説明しています。
- グリッドビュー内のJSON ドキュメント[475] は、グリッドビュー内のJSON ドキュメントの編集方法について説明しています。
- JSON スキーマビュー[505] は、ビューのJSON スキーマ編集機能およびJSON プロジェクトを作成する際での使用方法について説明しています。
- JSON データドキュメントの検証[537] は、JSON ドキュメント へのJSON スキーマの割り当てとJSON ドキュメントの検証について説明しています。
- JSON フラグメントの挿入[539] はJSON テキストフラグメントをJSON ドキュメントに外部ソースから素早く挿入する方法について説明しています。
- XSLT/XQuery を使用したJSON 変換[541] はXPath/XQuery 3.1 を使用してどのようにJSON ドキュメントをクエリするかについて説明しています。
- JSON のためのXQuery 式[543] はJSON ドキュメントを持つXQuery の使用方法について大まかに説明しています。
- JSON インスタンスからのJSON スキーマの生成[545] は、インスタンスからのスキーマの生成機能について説明しています。
- JSON スキーマからのJSON インスタンスの生成[548] はスキーマからインスタンスの説明方法について説明しています。
- JSON とXML 間の変換[549] はXMLSpy 内のJSON とXML 間の変換方法について説明しています。

13.1 JSON データ

このセクションは、どのように JSON データが構成されているかを簡単に説明します。JSON データは通常ドキュメントに保管されていますが、JSON (インスタンス) ドキュメントまたは他の型内でデータフラグメントとして保管されることもできます。JSON データフラグメントまたはドキュメントは、下で定義されているように JSON データ構造です。

XMLSpy は、ECMScript 5 拡張を追加する J SON の拡張である **JSON5** もサポートします。詳細に関しては、json5.org を参照してください。

JSON オブジェクトと配列

(通常ファイル拡張子 `.json` と共に保存される) JSON ドキュメント は、次のコアデータ構造により構築されます:

オブジェクト

オブジェクト は、中かっこで区切られており、順序のないゼロまたは `key:value` ペアのコレクションです。これらの `key:value` ペアは、オブジェクトのプロパティです。キーは常に文字列である必要があり、引用符で囲まれている必要があります。(プロパティ名と呼ばれる) キーは、コロンにより値から区切られています。プロパティ値は、JSON データ型 (*下のリスト参照* 465) のすべての値を使用することができます。プロパティはカンマで区切られています。下のリストは、(アトミック型の値を持つ) 3 つのプロパティを持つオブジェクトの例です:

```
{
  "emailtype": "home",
  "emailaddress": "contact01.home@altova.com",
  "citycode": 22
}
```

配列

配列 は、角かっこで区切られており、カンマで区切られたゼロまたは複数のアイテムのリストです。これらのアイテムは、JSON データ型のすべてを使用することができます (*下のリスト参照* 465)。

- *2 つのオブジェクトを含む配列*

  したの配列は、(それぞれ中かっこにより囲まれた) 2 つのオブジェクトから構成されています。配列自身は、角かっこにより表示されています。

```
[
      {
      "emailtype": "home",
      "emailaddress": "contact01.office@altova.com",
      "citycode": 22
      },
      {
      "emailtype": "office",
      "emailaddress": "contact01.office@altova.com",
      "citycode": 22
      }
  ]
```

- *オブジェクトのプロパティの値である配列*

  したのリストアは、3 つのキーを持つオブジェクトです。各値は、タプル (シーケンス) を含む配列です。(タプルは、ひとつの次元からなる配列と考えられることができます)。各タプル内の 3 つのアイテムは単純型です。

(C) 2015-2021 Altova GmbH

```
{
        "x": [ 1, 2, "abc" ],
        "y": [ 3, 4, "def" ],
        "z": [ 5, 6, "ghi" ]
}
```

JSON データ型

オブジェクトプロパティ値と配列アイテムは次の型であることができます:

- string（引用符で囲まれている必要があります）。文字列は date-time または email フォーマットなどの追加的にフォーマットを持つように指定されることができます。
- number: 小数点の部分を持つ数値。整数を含みます。
- integer: 小数点の部分を持たない数値。number 型のサブセット
- boolean（引用符で囲まれていない true/false）
- object: 他のオブジェクト内で使用されると、データをネストすることができます。
- array: オブジェクトにより許可されているより複雑な構造を作成する機能を与えます。
- null（引用符で囲まれていない null ）

JSON データのサンプル

下は、JSON データフラグメントのサンプルです。ドキュメントがどのようにオブジェクトと配列に構造化されているかに注意してください。また、キーの値のデータ型、引用内の文字列の値、他の型は緑に色付けされている点などに注意してください。

```
{
  "first": "Jason",
  "last": "Jones"
  "isManager": true,
  "age": 35,
  "address": {
    "street": "Jason Avenue",
    "city": "Jasonville",
    "state": "JS",
    "postcode": "JS12 ON34"
  },
  "phone": [
    {
      "type": "home",
      "number": "12 3456-7890"
    },
    {
      "type": "office",
      "number": "789 012-34567"
    }
  ],
  "children": [],
  "partner": null
}
```

JSON5 とJSON 間の違いについて

JSON5 は、JavaScript の厳密なサブセットあり、新規のJSON データ型を追加せず、全ての既存のJSON コンテンツと作業することができます。主要な違いは以下にリストされるとおりです:

- JSON5 はコメントをサポートします。コメントは以下のように区切ることができます: `// comment //` または `/* comment */`.
- JSON5 内では、**key:value** ペアのキーは、引用符で囲まれる必要はありません。
- JSON5 内では、文字列を複数のラインにわたり作成することができます。
- JSON5 ドキュメントを JSON スキーマに対して検証することができます。

(C) 2015-2021 Altova GmbH

13.2 JSON スキーマ

XML スキーマがXML ドキュメントの構造とコンテンツを指定するようにJSON スキーマはJSON ドキュメント内のJSON データが整理されるかを指定します。どのデータフィールドが期待されるか、また、値がどのように表示されるかを指定します。JSON スキーマ仕様とJSON スキーマについての情報はここで確認することができます。

JSON スキーマ自身は、JSON オブジェクトです。構文的には、スキーマ全体がJSON オブジェクトの区切りである中かっこで囲まれています（*下のリスト参照*）。スキーマはJSON 構文で書かれており、通常 `.json` の拡張子を持ちます。トップレベルオブジェクトの最初のキーワードである **`$schema`** キーワードを使用して、JSON スキーマとして表示されています。また以下の値を持つべきです: **`"http://json-schema.org/draft-N/schema#"`**。**`N`** がバージョンの番号です。`$schema` キーワードの使用方法のサンプルは以下のとおりです。

```
{
   "$schema": "http://json-schema.org/draft-04/schema#",
   ...
}
```

メモ　`$schema` キーワードは、スキーマの最新のバージョンを指定する以下の値を持つことができますが **`"http://json-schema.org/schema#"`**、特定のバージョンを指定するを使用することが奨励されます。. 詳細に関しては、JSON スキーマバージョン[506] を確認してください。

XMLSpy のJSON スキーマビュー内で、グラフィック性に富んだJSON スキーマを作成することができます。作成方法は、JSON スキーマビュー[505] に説明されています。スキーマ編集機能に加え、JSON スキーマビューで以下のスキーマに関連した機能を使用することができます:

- XMLSpy のJSON バリデータを使用して検証: JSON インスタントドキュメントにJSON スキーマを割り当て、XMLSpy 内から、インスタントドキュメントを検証します。 詳細に関しては、JSON ドキュメントの検証[537] を参照してください。
- JSON インスタンスからのJSON スキーマの生成[545]: JSON インスタントドキュメントが既存の場合、JSON スキーマを生成することができます。その後、必要に応じて、スキーマを編集することができます。
- JSON とXML の間の変換[549]: 2 つのフォーマットのドキュメント間での変換が行えます。

用語

GUI とこのドキュメント一般的に使用されるJSON スキーマ用語が以下にリストされています。

用語	定義
スキーマ	JSON スキーマドキュメント内のトップレベルのスキーマオブジェクトであるスキーマファイル
オブジェクト	ゼロまたは複数のプロパティを含むJSON 型
プロパティ	オブジェクトの `key:value`[464] ペア。値は任意のJSON データ型であることができます。
キーワード	オブジェクトの `key:value`[464] ペアの `key` 部分。常に文字列です。
サブスキーマ	演算子または依存性の子であるオブジェクト
定義	任意のJSON 型の完全な詳細。定義は、グローバルまたはローカル[512] であることができます。
配列	任意のJSON データ型のゼロまたは複数のアイテムのコンマにより区切られた順序リスト。
アトミック型	`string`, `number`, `integer`, `boolean`, および `null` JSON データ型
型セレクター	すべておよび複数型の選択する[527] `any` および `multiple` 型

演算子	子として定義に追加することのできる発生セレクター。*詳細に関しては、演算子* 530 *を参照してください。*

JSON データ型

オブジェクトプロパティ値と配列アイテムは次の型であることができます:

- `string`（引用符で囲まれている必要があります）。文字列は `date-time` または `email` フォーマットなどの追加的にフォーマットを持つように指定されることができます。
- `number`: 小数点の部分を持つ数値。整数を含みます。
- `integer`: 小数点の部分を持たない数値。`number` 型のサブセット
- `boolean`（引用符で囲まれていない `true/false`）
- `object`: 他のオブジェクト内で使用されると、データをネストすることができます。
- `array`: オブジェクトにより許可されているよりも複雑な構造を作成する機能を与えます。
- `null`（引用符で囲まれていない `null`）

(C) 2015-2021 Altova GmbH

13.3 JSON ラインと JSON コメント

XMLSpy はJSON ライン（JSONL） とコメント付きのJSON（JSONC）ドキュメントをサポートします。これはJSON ドキュメントの他の型のためにこれらのドキュメントの検証とインテリジェントな編集機能を使用することができることを意味します。このセクションではJSON ドキュメントのこれらの型の主要な機能について説明されています。

JSON ライン

JSON ライン（JSONL）は各レコードが次のレコードから新しいラインにより区分されている構造データを保管するためフォーマットです。この結果、レコードは一度に1つ処理され、ログファイルなどのデータを処理する際にこのようなフォーマットがとても役に立ちます。

Example JSON ライン document

```
["Team", "Played", "Won", "Drew", "Lost", "Points"]
["USA", 2, 1, 1, 0, 4]
["France", 3, 1, 1, 1, 4]
["Germany", 1, 0, 1, 0, 1]
["USA", 1, 0, 0, 1, 0]
```

サンプルに関しては、http://jsonlines.org/examples/ を参照してください。

JSON ラインファイルはファイルが `.jsonl` 拡張子を持つ場合、XMLSpy 内などで識別されます。.

コメント付きのJSON

other than JSON5 ドキュメント以外のJSON ドキュメントはコメントを許可していません。コメント付きのJSON フォーマット（JSONC）はJSON ドキュメント内でコメントを許可するように紹介されています。次のコマンドはJSONC 内で使用されており、XMLSpy 内でサポートされています:

- 単一ラインコメント: プレフィックス `//`. 例: `// My comment`
- 複数ラインコメント: 区切り文字 `/*` と `*/`. 例: `/* My comment */`

ファイルが `.jsonc` 拡張子を持つ場合、JSONC はそのままでXMLSpy 内で識別されます。i

メモ　JSON5 ドキュメント内でコメントは許可されています。（JSONC とJSON5 以外の）f JSON ファイルの他の型内でのコメントは処理中にエラーを引き起こす可能性があります。

13.4 JSON テキストビュー

Altova ウェブサイト: JSON エディター

JSON スキーマ、、および JSON/JSON5 インスタンスドキュメント場合、テキストビューの機能は以下を含みます: 折り返し余白[470]、構造マーキング[471]、構文色分け[471]、構文チェック[472] および自動補完[473]。XMLSpy はまた両方の方向の JSON/JSON5 と XML 間の変換[549] を行うことができ、JSON/JSON5 インスタンスからの JSON スキーマの生成[545]を可能にします。

折りたたみ余白

ソースの折りたたみは、JSON キーワードと定義に対して有効化され、ノードの展開および折りたたみを行う機能を意味します。このようなノードは +/- サインによりソースの折りたたみ余白内で表示されます（*下のスクリーンショット参照*）。余白は、テキストビュー設定ダイアログ[969] 内で切り替えオンとオフすることができます。ノードが折りたたまれると、視覚的に省略記号で表示されます（*下のスクリーンショット参照*）。マウスのカーソルが省略記号にポイントされると、折りたたまれたノードの内容がポップアップで表示されます（*スクリーンショット参照*）。内容がポップアップに大きすぎる場合、これは、ポップアップの下に省略記号で表示されます。

```
1   {
2       "Artists": [
3           {
4               "Name": "Queen",
5               "Albums": [
6                   {
7                       "Name": "A Night at the Opera",
8                       "Genre": [
9                           "Rock"
10                      ],
11                      "ReleaseDate": "1975-11-21",
12                      "Label": "EMI / Hollywood Records",
13                      "Tracks": [
14                          {
15                              "Title": "Death on Two Legs",
16                              "Duration": "03:43",
17                              "Writer": "Freddie Mercury"
18                          },
19                          {...},
24                          {...},
29                          "Title": "I'm in Love with My Car",
34                          "Duration": "03:05",
39                          "Writer": "Roger Taylor"
44                          {...},
49                          {...},
54                          {...},
```

テキストツーバー内の「全ての折りたたみを切り替える」アイコン は、全てのノードを展開された書式でまたはトップレベルのドキュメント要素まで折りたたみ表示します。
ノードの +/- アイコンをクリックすると以下のオプションを使用することができます:

クリック [-]	ノードを折りたたみます。
クリック [+]	子孫ノードが展開されて表示されるか、ノードが折りたたまれた前の状態に従い折りたたまれ、ノードが展開されます。
Shift+ [-]クリック	すべての子孫ノードを折りたたみ、展開フォーム内でクリックされたノードを残します。
Ctrl+ [+]クリック	クリックされたノードとそのすべての子孫ノードを展開します。

(C) 2015-2021 Altova GmbH

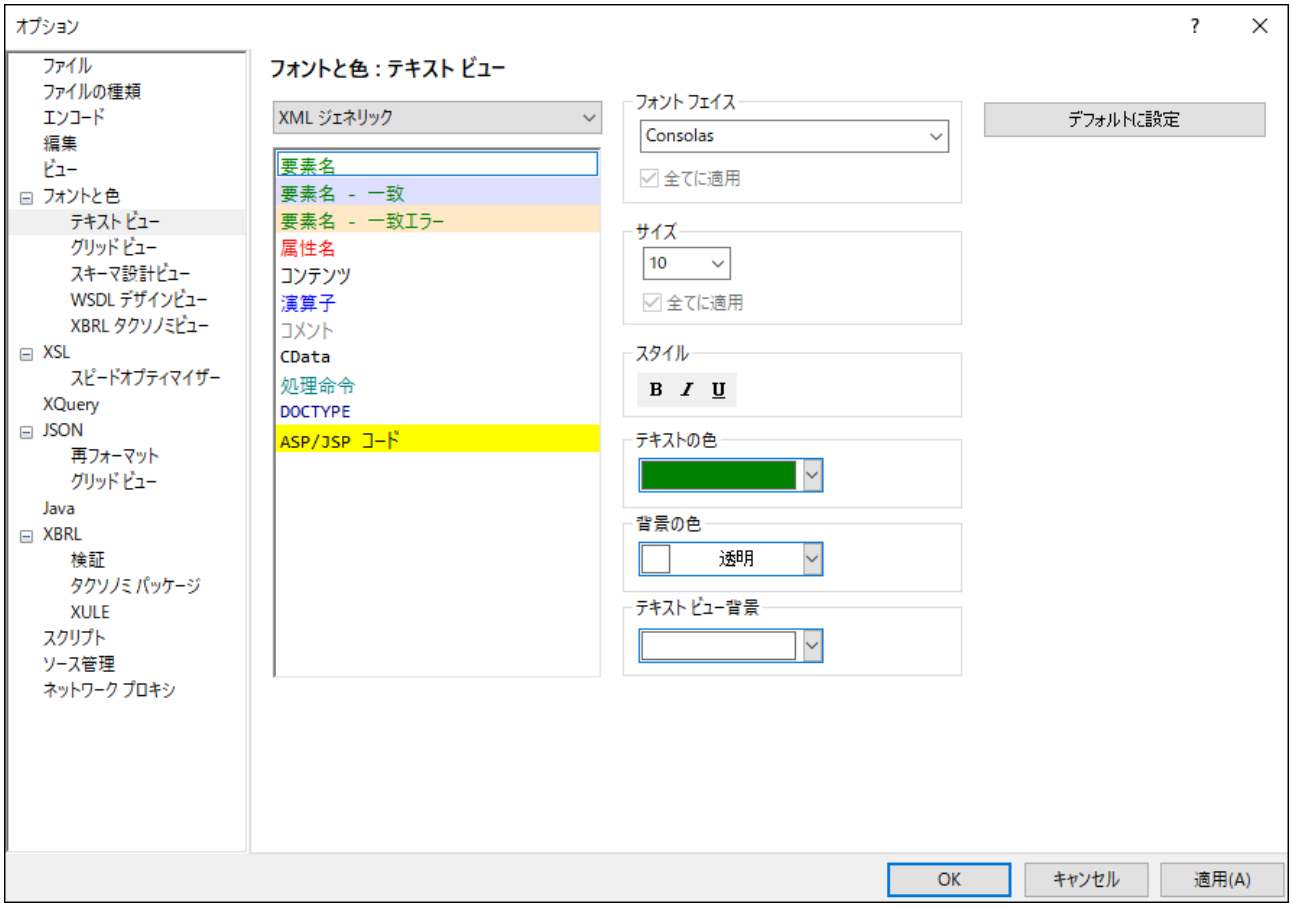
構造マーキング

JSON オブジェクトまたは配列の区切りを行う波かっこや角かっこ上にカーソルが位置している場合、対応するペアと共にかっこが太字で表示されます（*下のスクリーンショット*）。この機能により、特定の要素の開始点と終了点を簡単に確認することができます。

```
"Department": [ { "Name": "Administration",
  "Person": [ { "First": "Vernon",
    "Last": "Callaby",
    "Title": "Office Manager",
    "PhoneExt": 582,
    "EMail": "v.callaby@nanonull.com",
    "Shares": 1500,
    "LeaveTotal": 25,
    "LeaveUsed": 4,
    "LeaveLeft": 21},
```

構文による色分け

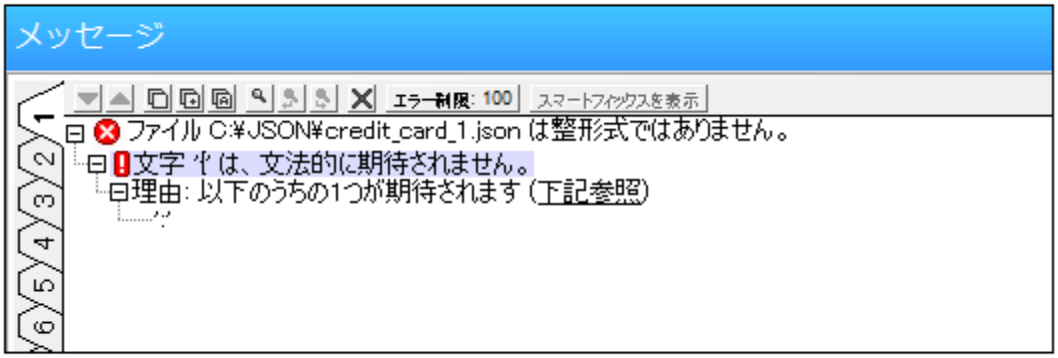
JSON ドキュメント（JSON *インスタンス/スキーマ*）、および JSON5 ドキュメントはオブジェクト文字列、値文字列、演算子、数値、そしてキーワードから構成されます。テキストビューでは、（下のスクリーンショットにあるように）オプションダイアログ[1014]の設定に従う形で（上のスクリーンショットで示されるように）各カテゴリーが異なる色で表示されます。オプションダイアログのテキストフォントタブでは、異なる JSON コンポーネントに対する色を指定することができます。左上のコンボボックスから JSON を選択し、各 JSON アイテムに対する目的の色を（スタイルペイン）から選択してください。



メモ JSON ではなく、JSON5 構文はコメントの使用を許可しています。JSON5 内のコメントは次のように区切ることができます: `// comment //` or `/* comment */`。

構文のチェック

[XML | **整形式のチェック**(F7)] コマンドを選択することで JSON ドキュメント (JSON *インスタンス/スキーマ*) の構文チェックを行うことができます。整形式チェックの結果は(以下のスクリーンショットにあるように)メッセージウィンドウに表示されます。



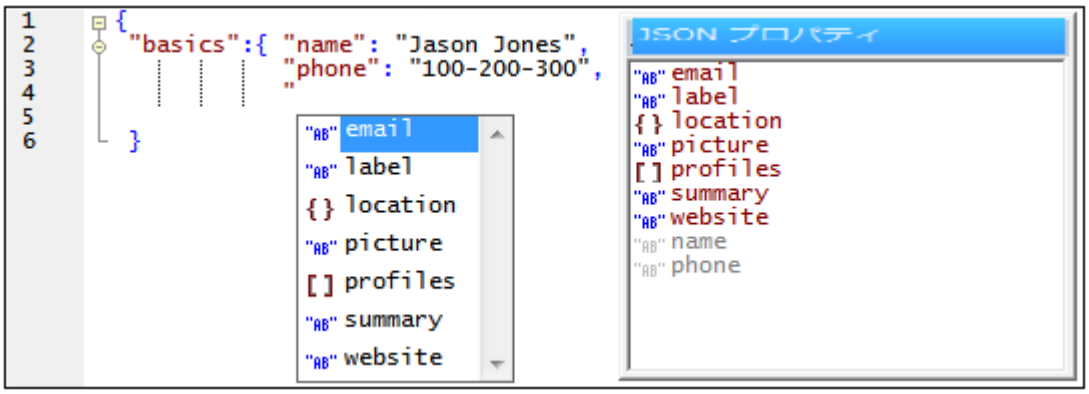
上のスクリーンショットで示されるエラーメッセージでは以下のことが指摘されています: コロンが必要な箇所で中かっこが存在します。

自動補完

自動補完は、編集される JSON ドキュメント（JSON インスタンス/スキーマ）がスキーマに関連する場合に有効化されます。

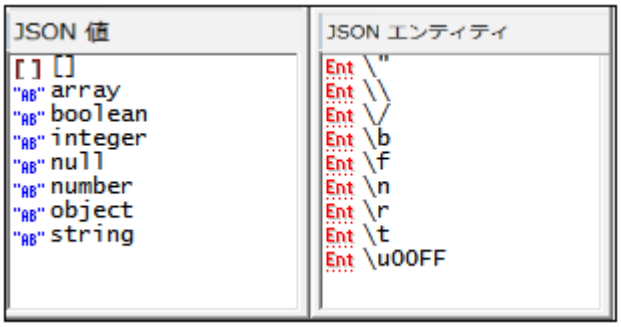
- ドキュメントが JSON スキーマの場合、自動補完は `$schema` 467 キーワードにより指定されているスキーマバージョンをベースにしています。詳細に関しては、JSON スキーマバージョン 506 を確認してください。
- ドキュメントが JSON/JSON5 インスタンスの場合、自動補完が有効化されるためには、JSON スキーマがインスタンスに割り当てられていなければなりません。 537

自動補完は、カーソルのロケーションで使用可能な入力オプションを提供します。これは以下により行うことができます（i）メインウィンドウ内のポップアップ、（ii）入力ヘルパーの使用（*下のスクリーンショット参照*）。ポップアップと入力ヘルパーはそれぞれ、カーソルのロケーションで有効な入力のリストを表示します。ポップアップリストのエントリを閲覧するには、矢印キーを使用して移動します。スキーマがエントリの詳細を含む場合は、詳細は（スキーマ内のエントリの説明 キーワード）ハイライトされたポップアップのエントリの横に表示されます。ポップアップウィンドウからエントリを選択または入力ヘルパー内のエントリをダブルクリックして挿入します。



上のスクリーンショットに表示されるインスタンスドキュメント内で、ポップアップと JSON プロパティ入力ヘルパーが、プロパティ名を示す引用符の後にカーソルがポイントされると表示されます。入力ヘルパーはこの時点で許可されているすべてのプロパティを表示し、既に入力されているプロパティは灰色で表示され無効化されています。ポップアップはその時点で使用することのできるプロパティを表示します。

他にも2つの入力ヘルパーがあります: JSON 値とJSON エンティティ（*下のスクリーンショット*）。これらはそれぞれ、文字列内のエスケープ文字のための許可されている `key:value` ペアとエンティティを表示します。JSON 値入力ヘルパーは、JSON スキーマを編集中に許可されている型キーワードを表示しています。JSON エンティティ入力ヘルパー `\u00FF` 内の最後の入力は、ユニコード文字をあらわすプレースホルダです。青でハイライトされた部分を使用するユニコード文字で置き換えることができます。



他のコンテキストを考慮した自動補完のエントリまたはヒントは、スキーマ内で指定されている場合、以下を含みます: 列挙、詳細、必要とされる発生とデフォルトの値。

(C) 2015-2021 Altova GmbH

13.5 JSON グリッドビュー

グリッドビューにより、JSON ドキュメント（JSON インスタンス/スキーマ）の構造をグリッドで確認することができ、構成されたテキストのブロックを再構成することができます。これにより、特に長く複雑に構成されたドキュメントの場合、テキストビューで必ず使用できるとも限らない、概要と編集機能が与えられます。JSON ドキュメントの場合、そのような複雑性は、他の配列とオブジェクトで複数のレベルにおよびネストされている配列とオブジェクトのフォームで出現する可能性があります。例えば、（テキストビューに表示されると同様）下にリストされる JSON テキストおよびグリッドビューでの表示を比較する場合（更に下のスクリーンショットで表示されています）。

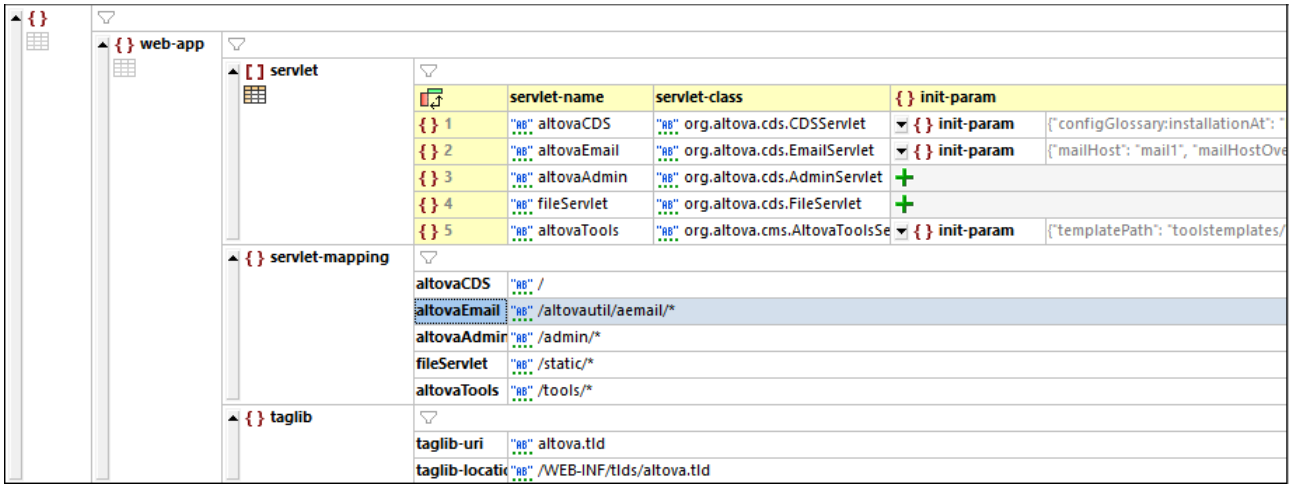
メモ　**Avro** は、Enterprise Edition のみでサポートされています。

テキストビュー内にリストされる JSON コード

```
{
  "web-app": {
  "servlet": [
    {
        "servlet-name": "altovaCDS",
        "servlet-class": "org.altova.cds.CDSServlet",
        "init-param": {
        "configGlossary:installationAt": "Philadelphia, PA",
        "configGlossary:adminEmail": "ksm@pobox.com",
        "configGlossary:poweredBy": "Altova",
        "configGlossary:poweredByIcon": "/images/altova.gif",
        "configGlossary:staticPath": "/content/static",
        "templateProcessorClass": "org.altova.WysiwygTemplate",
        "templateLoaderClass": "org.altova.FilesTemplateLoader",
        "templatePath": "templates",
        "templateOverridePath": "",
        "defaultListTemplate": "listTemplate.htm",
        "defaultFileTemplate": "articleTemplate.htm",
        "useJSP": false,
        "jspListTemplate": "listTemplate.jsp",
        "jspFileTemplate": "articleTemplate.jsp",
        "cachePackageTagsTrack": 200,
        "cachePackageTagsStore": 200,
        "cachePackageTagsRefresh": 60,
        "cacheTemplatesTrack": 100,
        "cacheTemplatesStore": 50,
        "cacheTemplatesRefresh": 15,
        "cachePagesTrack": 200,
        "cachePagesStore": 100,
        "cachePagesRefresh": 10,
        "cachePagesDirtyRead": 10,
        "searchEngineListTemplate": "forSearchEnginesList.htm",
        "searchEngineFileTemplate": "forSearchEngines.htm",
        "searchEngineRobotsDb": "WEB-INF/robots.db",
        "useDataStore": true,
        "dataStoreClass": "org.altova.SqlDataStore",
        "redirectionClass": "org.altova.SqlRedirection",
        "dataStoreName": "altova",
        "dataStoreDriver": "com.microsoft.jdbc.sqlserver.SQLServerDriver",
        "dataStoreUrl":
"jdbc:microsoft:sqlserver://LOCALHOST:1433;DatabaseName=goon",
        "dataStoreUser": "sa",
        "dataStorePassword": "dataStoreTestQuery",
        "dataStoreTestQuery": "SET NOCOUNT ON;select test='test';",
        "dataStoreLogFile": "/usr/local/tomcat/logs/datastore.log",
```

```
          "dataStoreInitConns": 10,
          "dataStoreMaxConns": 100,
          "dataStoreConnUsageLimit": 100,
          "dataStoreLogLevel": "debug",
          "maxUrlLength": 500
        }
        }, {
          "servlet-name": "altovaEmail",
          "servlet-class": "org.altova.cds.EmailServlet",
          "init-param": {
            "mailHost": "mail1",
            "mailHostOverride": "mail2"
          }
        }, {
          "servlet-name": "altovaAdmin",
          "servlet-class": "org.altova.cds.AdminServlet"
        }, {
          "servlet-name": "fileServlet",
          "servlet-class": "org.altova.cds.FileServlet"
        }, {
          "servlet-name": "altovaTools",
          "servlet-class": "org.altova.cms.AltovaToolsServlet",
          "init-param": {
            "templatePath": "toolstemplates/",
            "log": 1,
            "logLocation": "/usr/local/tomcat/logs/AltovaTools.log",
            "logMaxSize": "",
            "dataLog": 1,
            "dataLogLocation": "/usr/local/tomcat/logs/dataLog.log",
            "dataLogMaxSize": "",
            "removePageCache": "/content/admin/remove?cache=pages&id=",
            "removeTemplateCache": "/content/admin/remove?cache=templates&id=",
            "fileTransferFolder":
"/usr/local/tomcat/webapps/content/fileTransferFolder",
            "lookInContext": 1,
            "adminGroupID": 4,
            "betaServer": true
          }
        }
      ],
      "servlet-mapping": {
        "altovaCDS": "/",
        "altovaEmail": "/altovautil/aemail/*",
        "altovaAdmin": "/admin/*",
        "fileServlet": "/static/*",
        "altovaTools": "/tools/*"
      },
      "taglib": {
        "taglib-uri": "altova.tld",
        "taglib-location": "/WEB-INF/tlds/altova.tld"
      }
    }
  }
```

(*上にリストされる*) テキストビュー内のドキュメント構造を完成することは、時間をかけ慎重に確認する必要があります。(*下のスクリーンショットの*) グリッドビューでの構造は一目で確認することができます。



更に、構造は、簡単にオブジェクトを追加、削除、またはグリッド内でオブジェクトを移動させることで簡単に変更することができます。テキストの全体のブロックを(例えば、並べ替えまたは移動して)再編成することができます。コンテンツも同様にグリッドビューで編集することができます。これは個別のセル内でインセルコマンド[482]が使用できるようになるためです。

グリッドビューツールバー

アプリケーションウィンドウの上のツールバーエリアにあるグリッドビューツールバーを使用してグリッドビュー内の編集に関連するコマンドにアクセスすることができます(*下のスクリーンショット*)。名前とショートカットを確認するにはツールバーアイコンにマウスをポイントしてください。

メモ　これらのコマンドはJSON メニュー[848]内で使用可能です。

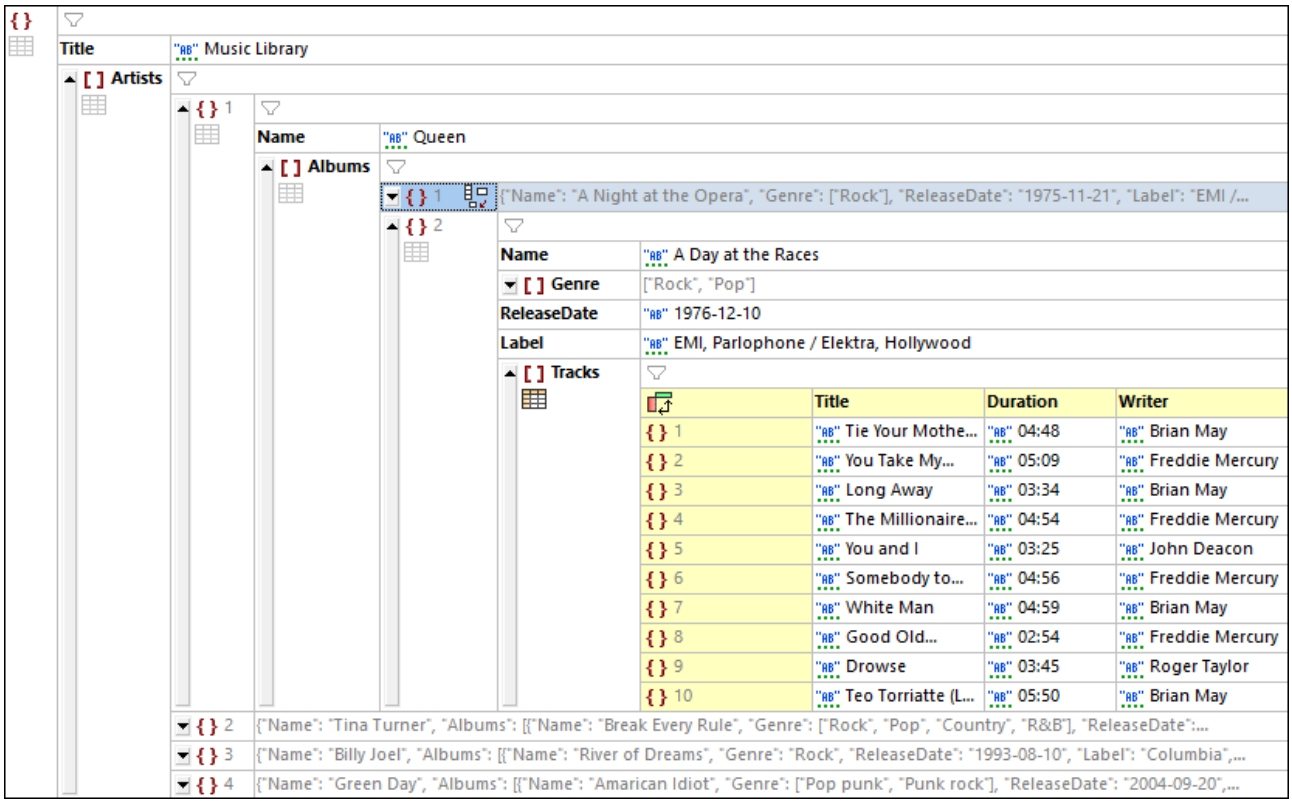
このセクション

このセクションは以下のリストに整理されています: JSON グリッドビュー固有の機能について説明されています。

- JSON ドキュメントがどのようにJSON グリッドビューで表示されるか[478]
- ドキュメント構造の編集[481]
- ドキュメントコンテンツの編集[482]
- グリッドビュー内でのドラッグとドロップ[485]
- テーブルの表示の使用[488]
- XQuery フィルターの使用[492]
- XQuery フォーミュラの使用[493]
- グリッドビュー内のイメージの表示[497]
- グリッドビュー内のコンテキストメニュー[500]
- グリッドビュー設定[502]

13.5.1 グリッドビュー内でのドキュメントの表示

JSON グリッドビューはグリッド内に JSON ドキュメントを階層的に表示します（*下のスクリーンショットを参照*）。



各グリッドラインは次のデータ構造を表示します。それぞれが下のテーブルでシンボルと共に表示されています。配列とオブジェクトの両方がオブジェクト、配列、または動的な値であることのできる子コンポーネントを含むことができます。

{ }	オブジェクト（*定義を参照してください*[464]）。オブジェクトには `key:value` ペアが含まれています。
[ ]	配列（*定義を参照してください*[464]）。配列には通常オブジェクトまたは値であるアイテムが含まれています。
	`key:value` ペア（*定義を参照してください*[464]）。

上のスクリーンショットでは、2つの `key:value` ペアを含むルートオブジェクトを確認することができます：(i) **`"Title":"Music Library"`**, (ii) **`"Artists":[Array]`**. **`Artists`** にキーされた配列にはオブジェクトである4つのアイテムが含まれています。

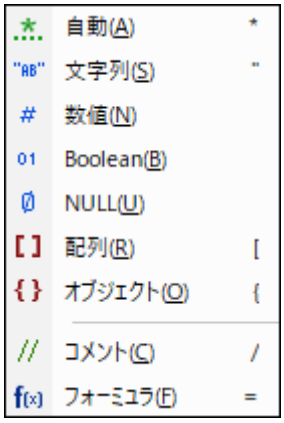
以下の点に注意してください：

- Ctrl + マウスのホイール または Ctrl+[プラス] と Ctrl+[マイナス] を使用してグリッドをズームすることができます。
- 同じレベルのオブジェクト、配列、配列アイテムは1から開始する番号が与えられています。配番されています。上のスクリーンショットでは、例えば、`Artists` 配列内ではオブジェクトは `1` から `4` に番号が付けられています。`Albums` と `Tracks` 配列内でも同様です。この番号づけは実際のドキュメント内には含まれておらず、ドキュメントの構造を確認するための JSON グリッドビュー機能です。

- オブジェクトと配列は構造のシンボルの左側の矢印アイコンを押して展開または退縮することができます *(上のスクリーンショットを参照)*。データ構造が退縮されると、そのコンテンツは単一のラインで表示されます。上のスクリーンショットの最後の3列を確認してください。
- 複数のコンポーネントを同じレベルで選択すると、これらを **Shift** を押し、選択されたコンポーネントの矢印をクリックして展開/折りたたむことができます。
- 配列とオブジェクトが展開されると以下に注意してください：(i) 配列とオブジェクトのコンテンツはシンボルの次のラインから開始します。(iii) 配列（キー）の名前は配列シンボルと同じラインに存在しますが、配列のアイテムは次のラインから開始します。配列またはオブジェクトが退縮されていると、コンテンツは同じラインに表示されます。
- テーブルディスプレイ 488 の一部である以外 `key:value` ペアは1つのラインに表示されることに注意してください（この場合キーは列ヘッダーになります）。テーブルディスプレイ 488 の例としては、上のスクリーンショット内の **`Tracks`** 配列のコンテンツを参照してください。

データ型（または型）のシンボル

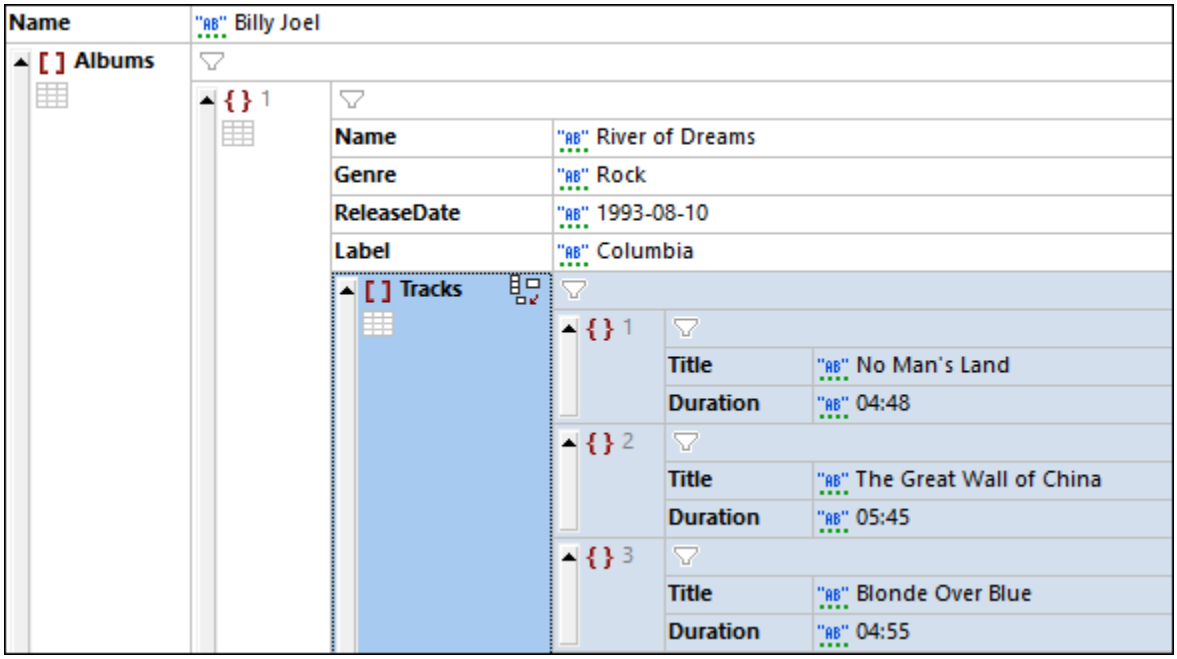
オブジェクトプロパティ値と配列アイテムは異なる型 465 であることができます。シンボルにより示されるそれぞれの型は下のスクリーンショットで示されています。（上のスクリーンショットでは、全ての `key:value` ペアの値の型は `string` 型であることに注意してください）。

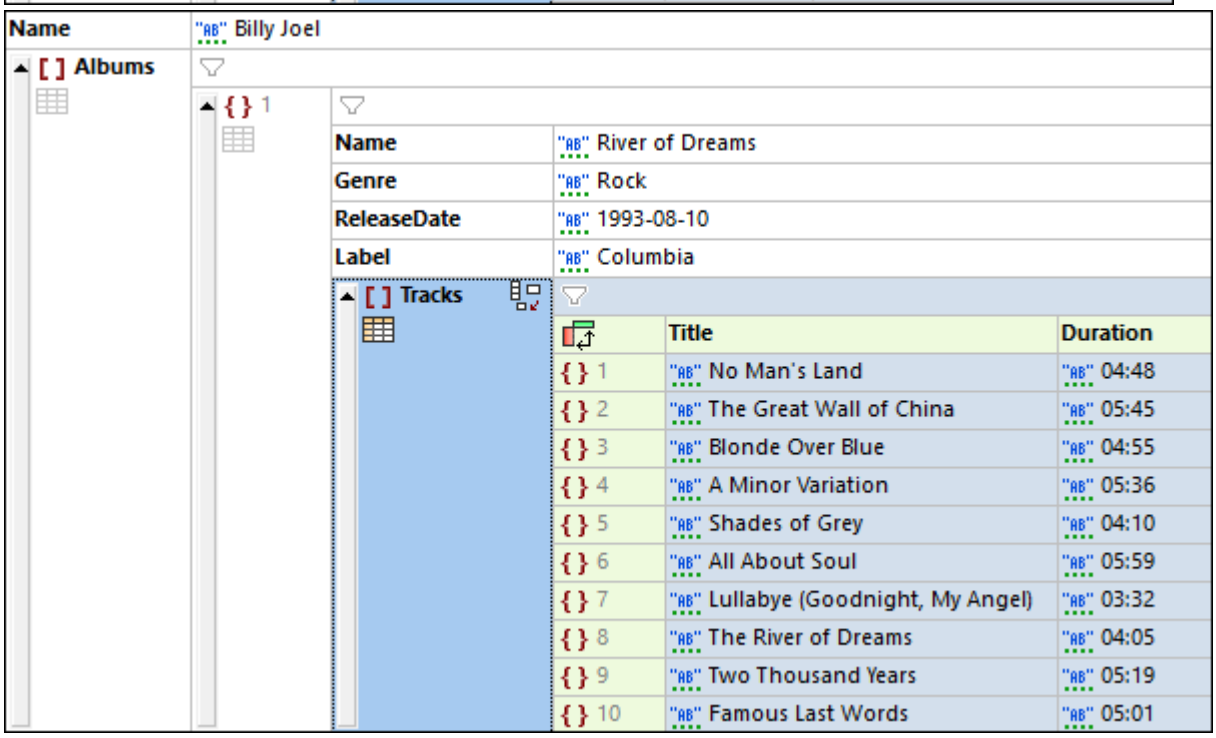


メモ　SONC とJSON5 内でコメントはサポートされています。フォーミュラ 492 はXMLSpy に固有です。

テーブルディスプレイ

少なくとも1つのオブジェクトまたは配列を含むオブジェクトと配列は*(左側のスクリーンショット内でハイライトされている)* リストまたは *(右側のスクリーンショット内でハイライトされている)* テーブルでビューすることができます。表示は個別のオブジェクトと配列のためにリストディスプレイとテーブルディスプレイの間を切り替えることができます。



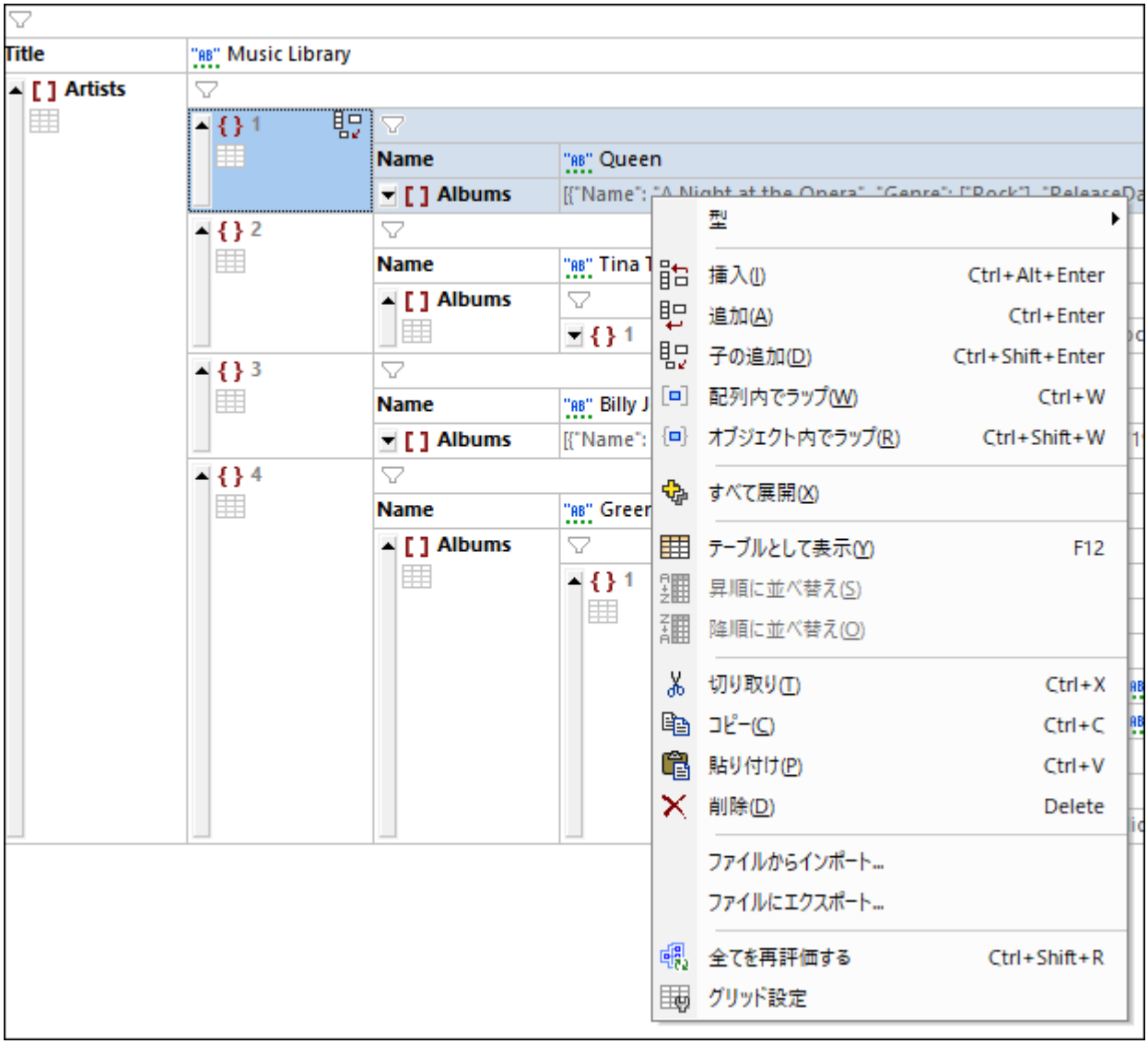


JSON グリッドビューのテーブルディスプレイの詳細に関してはセクション テーブルディスプレイ 488 を参照してください。

13.5.2 ドキュメント構成の編集

JSON ドキュメント構造を新規コンポーネントを追加、選択されたグリッドブロックを削除または移動してして編集することができます。ドキュメント構造を以下の方法で編集することができます:

- コンテキストメニュー内でコマンドを使用するためにコンポーネントを選択して右クリックします（*下のスクリーンショットを参照*）。セクション コンテキストメニュー 500 でコマンドについて説明されています。
- このセル内で使用可能な編集ボタンをします。これらのボタンについては下のセクション セル内のコマンドボタン 482 で説明されています。
- コンポーネントを選択してドラッグアンドドロップを使用して新規のロケーションにドロップし構造を変更します。



構造を作成するためのコマンドのショートカットは下のテーブルにリストされています。

コマンド	ショートカット

後に（兄弟）を挿入	Ctr+Enter
前に（兄弟）を挿入	Ctr+Shift+Enter
付加（兄弟）	Ctr+E
子の追加	Ctr+Alt+Enter
配列内でラップ	Ctr+W
オブジェクト内でラップ	Ctr+Shift+W

メモ　プロパティとしてコンポーネントが追加（挿入、付加、子として追加）され、型をオブジェクトまたは配列に変更する場合、これをコンポーネントの型シンボルをクリックして型を選択します 482 。

セル内のコマンドボタン

JSON グリッドビュー内では、グリッドのセル内のボタンは対応するセルのコンテキスト内で使用可能な機能を提供します。これらのボタンコマンドの使用は一般的に使用される操作を更に素早く簡単に実行する方法です。使用可能なセル内のボタンコマンドは下のテーブルにリストされています。表示に関連したコマンドは青い背景と共に表示されています。ドキュメント（構造とコンテンツ）の編集のためのコマンドはベージュの背景で表示されています。特定のセルのコンテキスト内の追加コマンドのためにセルのコンテキストメニュー 500 を使用してください。メモ ボタンをポイントするとセルボタンのコマンドの名前はツールのヒントとして表示されます。

	セルコンテンツに従いグリッド列の幅を最適化
/	色で表示されている場合 テーブルの表示 488 がオンに切り替えられています。それ以外の場合はオフです。表示するためにクリックします。
/	テーブルの左側のセル内のコマンドをトグルします。行を列に、または列を行に切り替えます。
/	色で表示されている場合、テーブル、オブジェクト、または配列のためのフィルター 492 が有効化されています。それ以外の場合フィルターは無効化されています。無効化/有効化するためにクリックします。式を編集するには、ダブルクリックします。
	セルをワードラップするためのトグルコマンド。選択されると、ワードラップがオンになります。デフォルトでは、アイテムのコンテンツのみがラップされます。アイテムの名前を追加でラップする場合、このオプションをアイコンのドロップダウンリストから選択します。
	配列内: リストまたはテーブルに配列アイテムをふかします。 オブジェクト内: `key:value` ペア（をリスト内で、または（テーブル表示で）新規のテーブル列のセルとして）付加します。
	`key:value` ペアを追加します。デフォルトでは値の型は `string` です。
	プロパティ値のデータ型を選択、またはコメントまたはフォーミュラを入力します。

メモ　他のツールバーコマンドの詳細に関しては、セクション グリッドビュー内のコンテキストメニュー 500 を参照してください。

13.5.3 ドキュメントコンテンツの編集

JSON グリッドビュー内のコンテンツの編集はとても理解しやすいです: 編集するセルの内部をダブルクリックして必要に応じてコンテンツを編集します。更に JSON グリッドビューは以下の編集機能を提供します:

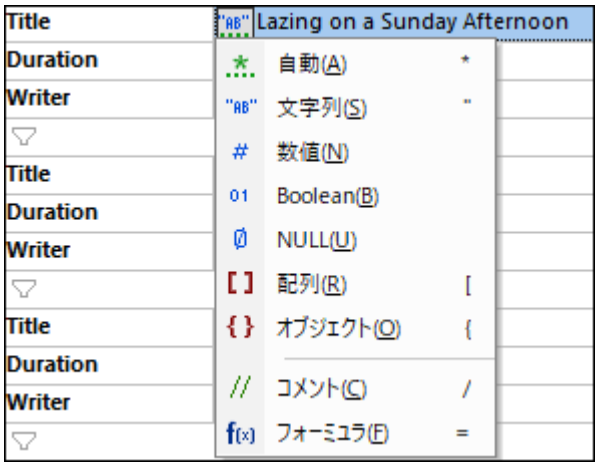
- 型選択機能および自動型検知と値の変換機能
- JSON スキーマをベースにした JSON ドキュメント内の自動補完

型の自動検知の型選択

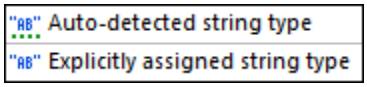
JSON グリッドビューは次の型のカテゴリを区別します:

- *単純型:* String、Number、Boolean、Null
- *特別な単純型:* (値から検知される) Auto、Comment (JSONC、JSON5)
- *コンテナー型:* Object、Array
- *特別のコンテナー型:* フォーミュラ

値の前のシンボルにより示される値の型 (*下のスクリーンショットを参照*)。型を変更するには、シンボルをクリックして、表示されるメニューから型を選択します。型のシンボルとショートカットは下のスクリーンショットに表示されています:



自動検知される型は緑の点線が下に表示されています (*下のスクリーンショットを参照*)。明示的な型は割り当てられる型です。



型に関連したアクションは2 つのシチュエーションで発生します:

- JSON ドキュメントがロードされる場合: 全ての単純型は値から自動的に検知された `Auto` に変換されます。例えば、`"MyString"` は自動的に文字列型、`123` が数値型として、`true` がブール型として、および `null` が Null 型として検知されます。曖昧な文字列の場合、型を明示的に選択してください。
- データ構造、または値が入力された場合: 単純型は自動的に検知され型は自動的に割り当てられます。型を後に変更することができます。

メモ　JSON グリッドビュー設定 502 により以下を指定することができます (i) 複数のセルが選択されている場合型の変更がどのように処理されるかを指定。(ii) 型が配列またはオブジェクトに変更される場合、動的な型の値がどのように扱われるかを指定。

自動補完

編集中の自動補完は JSON ドキュメント (JSON インスタンス/スキーマ) がスキーマに関連付けられている場合有効化されます。

- ドキュメントがJSON スキーマの場合、自動補完は `$schema` 467 キーワードにより示されるスキーマバージョンをベースにします。詳細に関しては、JSON スキーマバージョン 506 を参照してください。
- ドキュメントがJSON/JSON5 インスタンスの場合、自動補完を有効化するには、JSON スキーマがインスタンスに割り当てられている必要があります 537 。

自動補完はカーソルの位置で使用することのできる入力オプションを提供します。これは以下を使用して行うことができます(i) メインウィンドウ内のポップアップを使用して(ii) 入力ヘルパーを使用して(*下のスクリーンショットを参照*)。ポップアップと入力ヘルパーはそれぞれカーソルのロケーションで有効なエントリのリストを表示します。ポップアップリスト内のエントリを移動するには矢印キーを使用してください。エントリをポップアップウィンドウから選択、または入力ヘルパー内のエントリをダブルクリックして挿入します。

セル内のコマンドボタン

JSON グリッドビュー内では、グリッドのセル内のボタンは対応するセルのコンテキスト内で使用可能な機能を提供します。これらのボタンコマンドの使用は一般的に使用される操作を更に素早く簡単に実行する方法です。使用可能なセル内のボタンコマンドは下のテーブルにリストされています。表示に関連したコマンドは青い背景と共に表示されています。ドキュメント(構造とコンテンツ)の編集のためのコマンドはベージュの背景で表示されています。特定のセルのコンテキスト内の追加コマンドのためにセルのコンテキストメニュー 500 を使用してください。メモ ボタンをポイントするとセルボタンのコマンドの名前はツールのヒントとして表示されます。

	セルコンテンツに従いグリッド列の幅を最適化
/	色で表示されている場合 テーブルの表示 488 がオンに切り替えられています。それ以外の場合はオフです。表示するためにクリックします。
/	テーブルの左側のセル内のコマンドをトグルします。行を列に、または列を行に切り替えます。
/	色で表示されている場合、テーブル、オブジェクト、または配列のためのフィルター 492 が有効化されています。それ以外の場合フィルターは無効化されています。無効化/有効化するためにクリックします。式を編集するには、ダブルクリックします。
	セルをワードラップするためのトグルコマンド。選択されると、ワードラップがオンになります。デフォルトでは、アイテムのコンテンツのみがラップされます。アイテムの名前を追加でラップする場合、このオプションをアイコンのドロップダウンリストから選択します。
	配列内: リストまたはテーブルに配列アイテムをふかします。 オブジェクト内: `key:value` ペア(をリスト内で、または(テーブル表示で) 新規のテーブル列のセルとして)付加します。
	`key:value` ペアを追加します。デフォルトでは値の型は `string` です。
	プロパティ値のデータ型を選択、またはコメントまたはフォーミュラを入力します。

メモ 他のツールバーコマンドの詳細に関しては、セクション グリッドビュー内のコンテキストメニュー 500 を参照してください。

(C) 2015-2021 Altova GmbH

変更中に検証

編集中に検証 モードはデフォルトでオンに切り替えられています。オンに切り替えられていると、JSON グリッドビュー内でのドキュメントの変更中に整形式チェックと検証チェックが実行されます。(整形式のチェックに加え) JSON ドキュメントの検証を実行するには、JSON スキーマが JSON ドキュメントに割り当てられている必要があります。エラーはエラーの存在するテキストを赤色で表示し、エラーの箇所を赤い感嘆符で表示することで表示されています。詳細に関しては JSON ドキュメントの検証 537 を参照してください。

編集中に検証 モードは (i) 「XML | 編集中に検証」 847 メニューコマンド (ii) **編集中に検証** ツールバーボタン または (iii) オプションダイアログの検証設定 1015 の *編集時* オプションによりオンとオフに切り替えることができます。

検索と置換

JSON グリッドビュー内でテキストを検索し置き換えるために「**検索**」(Ctrl+F) と「**置換**」(Ctrl+H) コマンドを編集メニューで使用することができます。結果は含まれるコンポーネントもオレンジ色でハイライトされた状態でオレンジ色でハイライトされています。長いテキスト文字列を含むセルの場合、セルのみがハイライトされています。

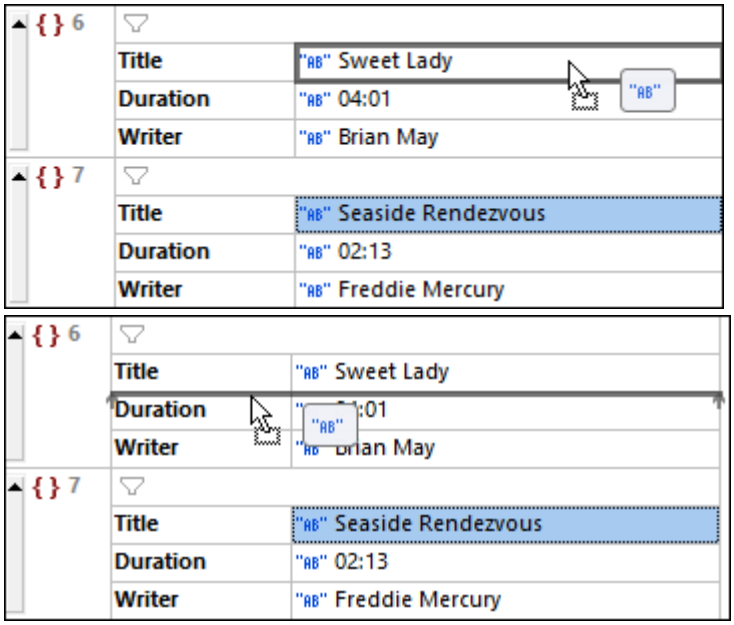
13.5.4 グリッドビュー内でのドラッグとドロップ

グリッドビューは JSON ドキュメントフラグメントをグリッドビューに XMLSpy 内、外部アプリケーション、または Web サイトで開かれているドキュメントからドラッグすることのできる役に立つドラッグオーバーレイを提供します。

ターゲットノードにカーソルをポイントすると、ドラッグされているアイテムだけでなくドロップされた際にどのようにアイテムが作成されるかについての情報を提供するドラッグオーバーレイが表示されます。例えば、下のスクリーンショットでは `Seaside Rendezvous` がドラッグ (移動) されています。アイテムが文字列であることをが `AB` ポップアップにより示されています。

- (左側のスクリーンショットで示されている通り) 値フィールドにポイントすると、フィールド全体がマークされ、マークされたフィールドの値がドラッグされたフィールドの値と置き換えられます。
- (右側のスクリーンショット内の `Duration` などの) ノード名にポイントすると、その箇所にドロップされる値文字列が値キーペアの値ノードを作成することを示す挿入ラインが表示されます。この場合のキーは空の文字列になります。

メモ　**Ctrl+Z** または メニューコマンド **Edit | Undo** を使用して移動を元に戻すことができます。

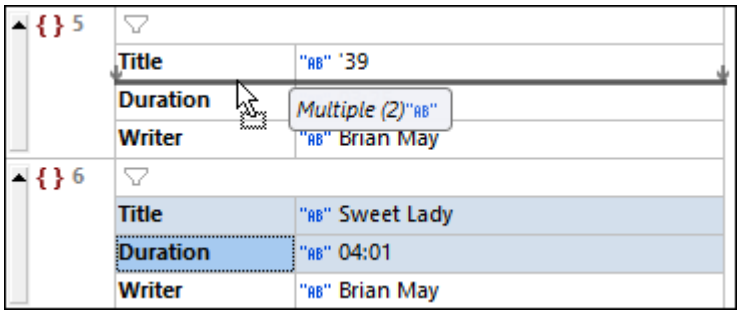


ドラッグオーバーレイ内に含まれている情報

以下の種類のドラッグオーバーレイ情報が表示されます。

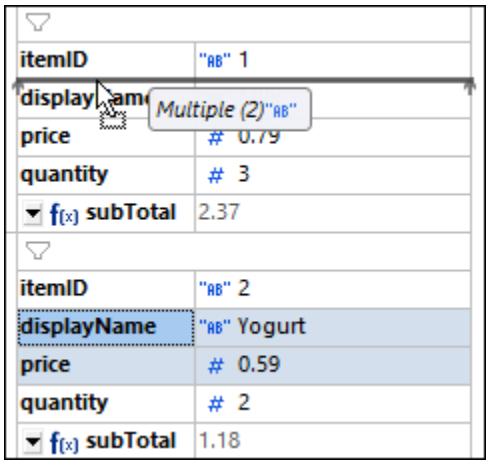
通常のグリッドビュー

- 値フィールドにドロップされた値はターゲットフィールド内の値を上書きするためにソース値を移動します（*左上のスクリーンショット参照*）。
- ツリーにノードとしてドロップされる値はキー値ペアの値ノードを作成します（*右上のスクリーンショット参照*）。キーは空の文字列になります。
- 単純型の複数のノードのオーバーレイはドロップされたノード型とこれらのノードの数値を表示します。（値ではなく）ノードを選択するために、ノードの名前をクリックする必要があります。下のスクリーンショットでは、ドラッグオーバーレイ内の情報は2個の型文字列のキー値ペアがドロップされることを示しています。



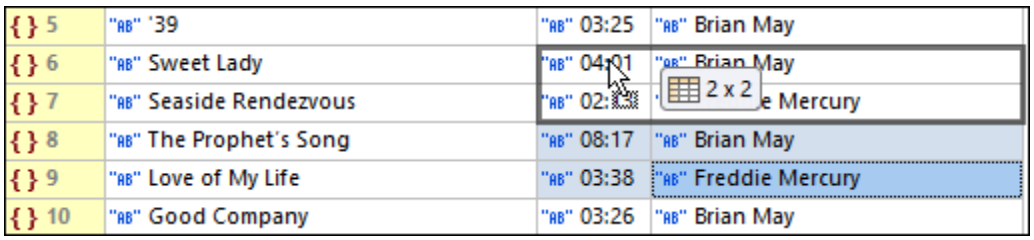
- 異なる型の複数のノードのためのオーバーレイは新規のノードが文字列のデフォルトの型と共に作成されることを表示しています（*下のスクリーンショット参照*）。対応する *型の変更* アイコンをクリックすることでドロップされたノードの型を変更することができます。（値ではなく）ノードを選択するために、ノードの名前をクリックする必要があります。

(C) 2015-2021 Altova GmbH



テーブルの表示

- 値がドラッグされると、選択されたセルの数値は 列 x 行 により示されています（*下のスクリーンショット参照*）。ドラッグされたセルの間トリックは対応するマトリックスを置き換えます。ターゲットセル（マトリックスをドロップするセル）はマトリックスの左のセルの値を受け取ります。他のドラッグされたセルがターゲットセルから右方向および下方向にフィルします。例えば、下のスクリーンショットでは青色のセルがドラッグされ行 6 の`Duration` セルにドラッグされています。これによりマークされたセルが置き換えられます。ドラッグされたセルのマトリックスがテーブルの枠を超えると、列および/または行の適切な数量がすべてのドラッグされたセルに対応するように追加されます。この場合、拡張される枠は点線で表示されます。



- ターゲットがセルではなくノードツリーの場合、追加されるノードは(i)（角かっこに含まれる）追加されるノード、と(ii) 新規のアイテムのインスタンスの数量と共に表示されます。例えば、下のスクリーンショットでは、2つのオブジェクト `Title`、`Duration` と `Writer` キー値ペアを含む配列が追加されます。

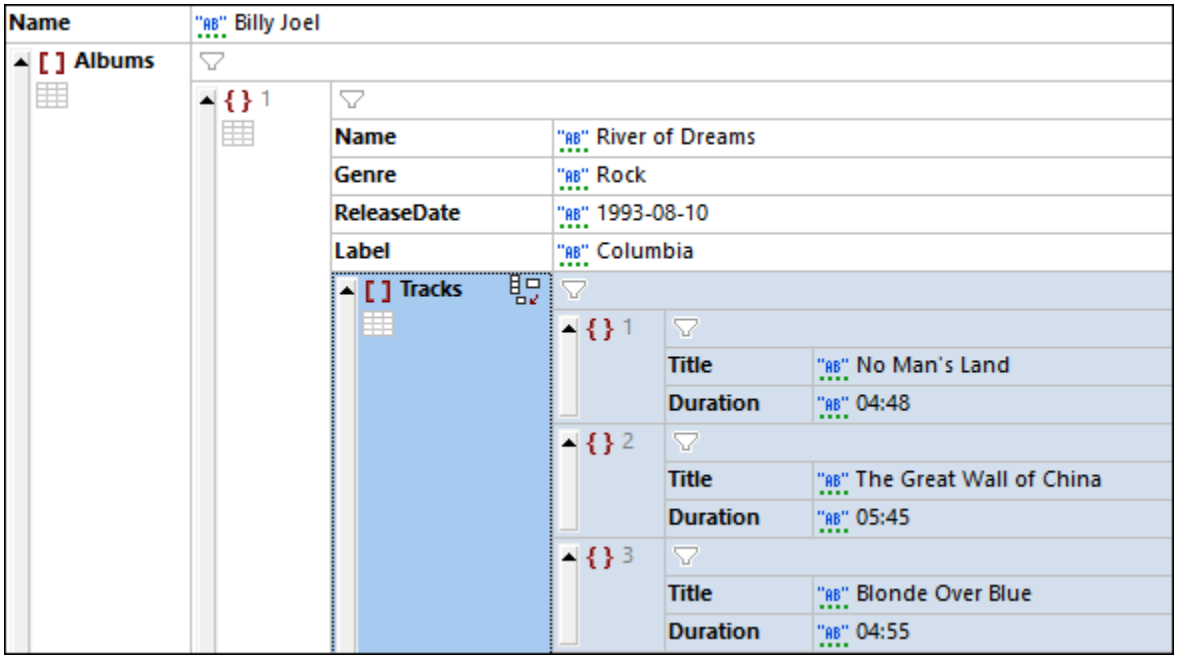
- （ヘッダーをクリックして）選択し、隣接する列ヘッダーにドロップして列を移動することができます。挿入ラインは移動された列が配置されるターゲット列のサイドを示しています。

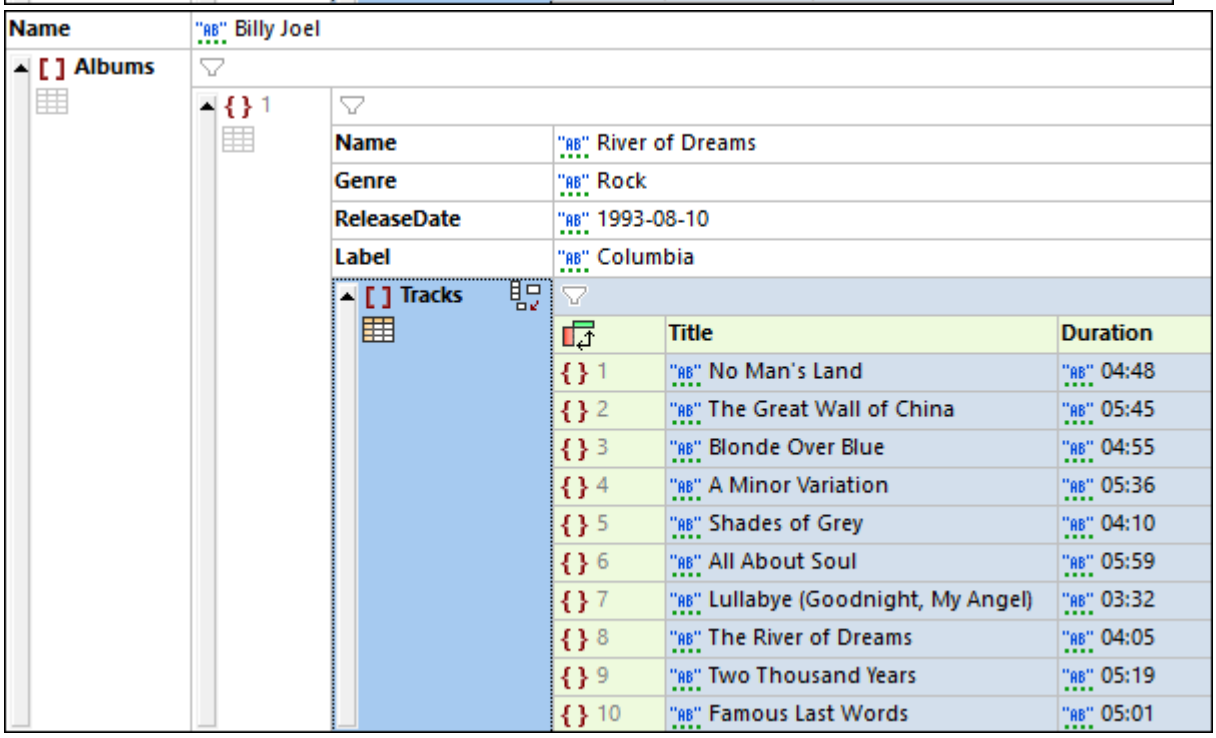
13.5.5 テーブルの表示

テーブルの表示ビュー

少なくとも1 つのオブジェクトまたは配列を含むオブジェクトと配列は*(左側のスクリーンショット内でハイライトされている)* リストまたは *(右側のスクリーンショット内でハイライトされている)* テーブルでビューすることができます。これら2 つのビュー間を切り替えるには、オブジェクトまたは配列アイコンの下にあるテーブルアイコンをクリックします *(下のスクリーンショット参照)*。配列テーブル内で、配列アイテムは行として表示されます。例えば、右下のスクリーンショットではトラック配列がテーブルとして表示されています。この配列は行として表示される子オブジェクトにより構成されています。各オブジェクトのプロパティ（下のスクリーンショット内の `Title` と `Duration`）は列のヘッダーとして表示されている共通のプロパティの名前と共に列として表示されています。行と列を切り替えるには、テーブルの左上のセル内のアイコンをクリックします。

(C) 2015-2021 Altova GmbH





テーブル表示内のビューと編集のためのアイコン

下に表示されるアイコンはテーブル表示内で使用可能なアイコンで、ビューと編集機能を提供します。

	セルコンテンツに従いグリッド列の幅を最適化

/	色で表示されている場合 テーブルの表示 488 がオンに切り替えられています。それ以外の場合はオフです。表示するためにクリックします。
/	テーブルの左側のセル内のコマンドをトグルします。行を列に、または列を行に切り替えます。
/	色で表示されている場合、テーブル、オブジェクト、または配列のためのフィルター 492 が有効化されています。それ以外の場合フィルターは無効化されています。無効化/有効化するためにクリックします。式を編集するには、ダブルクリックします。
	セルをワードラップするためのトグルコマンド。選択されると、ワードラップがオンになります。デフォルトでは、アイテムのコンテンツのみがラップされます。アイテムの名前を追加でラップする場合、このオプションをアイコンのドロップダウンリストから選択します。
	配列内: リストまたはテーブルに配列アイテムをふかします。 オブジェクト内: `key:value` ペア(をリスト内で、または(テーブル表示で) 新規のテーブル列のセルとして)付加します。
	`key:value` ペアを追加します。デフォルトでは値の型は `string` です。
	プロパティ値のデータ型を選択、またはコメントまたはフォーミュラを入力します。

テーブル表示内の編集

テーブルの表示では、以下にリストされる編集アクションを実行することができます。

テーブルに行を追加する

以下の方法で行を追加することができます:

- *テーブルに子行を追加します:* テーブルを選択してコンテキストメニュー内でコマンド **子の追加** (Ctrl+Shift+Enter) を選択します。または、テーブルの **子の追加** アイコンをクリックします(*上のアイコンリストを参照*)。新しい行は既存のテーブルの下に付加されます。
- *行が選択された際に兄弟行を追加します:* 行を選択してコンテキストメニュー内でコマンド **追加** (Ctrl+Enter) を選択します。新しい行は既存のテーブルの下に付加されます。
- *選択された行の上に兄弟行を追加します:* 行を選択してコンテキストメニュー内でコマンド **挿入** (Ctrl+Alt+Enter) を選択します。

プロパティの値の入力と編集

プロパティ値が存在するテーブルセルを選択し値を入力します。値が文字列、数値、ブール値または null の場合、データベースは自動的に検知されます。曖昧な場合、型は 文字列 に設定されます。自動選択をデータ型選択アイコンを使用してオーバーライドすることができます(*上のアイコンリストを参照してください*)。または、適切なショートカットを使用することができます(*下のショートカットテーブルを参照してください*)。

*	自動(文字列、番号、ブール値、null を検知し適宜設定します)
"	文字列
[	配列
{	オブジェクト
/	コメント(ドキュメントは JSONC または JSON5 である必要があります)
=	フォーミュラ(ドキュメントは JSONC または JSON5 である必要があります)

新規の列に内に空の行を追加します。
行に新規のセルを追加することができます。新規のセルは新規に作成された列の一部になります。新規のセルを追加するには行を選択して、コンテキストメニュー内でコマンド**子の追加** (Ctrl+Shift+Enter)を選択します。または、行の**行セルの追加** アイコンをクリックします*(上のアイコンリストを参照)*。新規の列内に新規の行セルが作成されます。セル内の値の型はデフォルトで 文字列 です。プロパティ値をセル内に、プロパティの名前を列ヘッダーとして入力します。新規に作成された列内の他のセルは空です。

選択された列の値のテーブル行を並べ替える
自身の子ノード(列)の一つの相対的な値の上のテーブルの行を並べ替えることができます。例えば、アルファベット順に並べ替えられたテーブルの繰り返し要素を与えるために*LastName* 列をベースに行を並べ替えることができます。列を並べ替えるには、列ヘッダーを選択し、グリッドビューツールバー[477] コマンド **昇順で並べ替え** または **降順で並べ替え**をクリックします。これらのコマンドはJSON メニュー[851] 内でも使用可能です。

テーブル表示と外部アプリケーション

テーブル表示と(MS Excel などの)スプレッドシートアプリケーション間でデータを転送するためにテーブル構造を活用することができます。データの表示からデータを移動するには、以下を行います:

1. テーブルの表示内で、コピーするノードを選択します。セル自身、列ヘッダー、行ヘッダーまたはテーブル全体をクリックしてこれを行います。テーブル全体または列ヘッダーをクリックすると、列ヘッダーもコピーされます。それ以外の場合はコピーされません。(下のスクリーンショットでは行ヘッダーと共に行 1 から 8 が選択されています。)

[ ] Tracks	Title	Duration	Writer
{ } 1	"AB" Death on Two Legs	"AB" 03:43	"AB" Freddie Mercury
{ } 2	"AB" Lazing on a Sunday Afternoon	"AB" 01:08	"AB" Freddie Mercury
{ } 3	"AB" I'm in Love with My Car	"AB" 03:05	"AB" Roger Taylor
{ } 4	"AB" You're My Best Friend	"AB" 02:50	"AB" John Deacon
{ } 5	"AB" '39	"AB" 03:25	"AB" Brian May
{ } 6	"AB" Sweet Lady	"AB" 04:01	"AB" Brian May
{ } 7	"AB" Seaside Rendezvous	"AB" 02:13	"AB" Freddie Mercury
{ } 8	"AB" The Prophet's Song	"AB" 08:17	"AB" Brian May

2. コンテキスト コピー| タブで区切られたテキストとしてコピーする[764] を選択します。
3. データを直接スプレッドシートプログラムに張り付けます。

	A	B	C
1	Title	Duration	Writer
2	Death on Two Legs	3:43	Freddie Mercury
3	Lazing on a Sunday Afternoon	1:08	Freddie Mercury
4	I'm in Love with My Car	3:05	Roger Taylor
5	You're My Best Friend	2:50	John Deacon
6	'39	3:25	Brian May
7	Sweet Lady	4:01	Brian May
8	Seaside Rendezvous	2:13	Freddie Mercury
9	The Prophet's Song	8:17	Brian May

データの転送は両方向に行うことができます。スプレッドシートからテーブルの表示にデータをコピーすることができます。以下の手順を行ってください:

1. 外部アプリケーション内で範囲を選択して（**Ctrl+C** を使用して）クリップボードにコピーします
2. テーブルの表示内で単一のセルを選択します。
3. コピーされたデータを **Ctrl+V**を使用して貼り付けます。

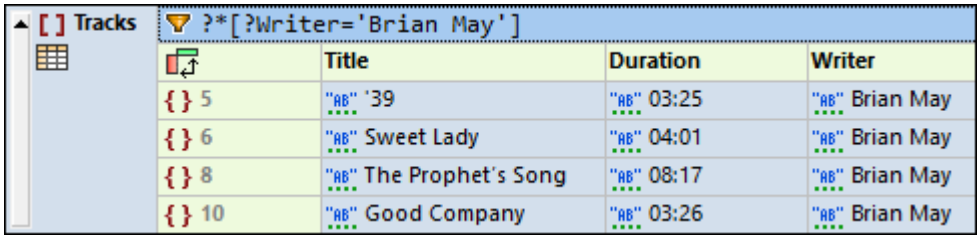
データは元の構造に対応し、テーブルビュー内で選択されたセルから開始する構造を持つテーブルに張り付けられます。以下の点に注意してください。

- 既存のデータを無効化することなく新規の行および/または列の作成を許可するセルにデータがコピーされていると、新規の行および/または列が作成されます。
- 列はテーブル（行）のオブジェクトのプロパティとして作成されますが、行は新規のオブジェクトとして作成されます。
- テーブル構造が有効に変更できない場合、新規のデータはテキストとして選択されたセルの元のデータを上書きします。

更に複雑なデータ転送タスクのために、XMLSpy のユニークな インポート/エクスポート 938 関数を使用してください。

13.5.6 フィルター

フィルターはオブジェクトと配列をフィルターし表示することを可能にします。例えば、下のスクリーンショットでは（XQuery 3.1 式である）フィルターが配列に適用され、Brian May により作成されたトラックのみが表示されます。JSON では、文字列値 `Brian May` を含む **`Writer`** プロパティを持つ **`Tracks`** のオブジェクト子のみが表示されます。）フィルターの XQuery 式は **`Tracks`** のすべての子オブジェクトをルックアップし、文字列 `'Brian May'` に一致する **`Writer`** プロパティをルックアップするものを選択します。



各フィルターは個別に実行されドキュメント内の他のフィルターまたはフィルターにより影響されません。

JSON のための XQuery 式ドキュメントの作成に関する詳細は、JSON のための XQuery 式 543 を参照してください。

メモ グリッドビュー内にフィルターとフォーミュラのために式を入力する際に、読みやすいように式を複数行で表示するようにすることが奨励されます。式内に新規のラインを追加するには、「Ctrl+Enter」を押します。

フィルターのセットアップ

フィルターをセットアップするには、フィルターする要素を右クリックして **フィルター** を表示されるコンテキストメニューから選択し Xquery 式を入力して **入力**をクリックします。*フィルターセルはアイコン* ▼ により示されています*（上のスクリーンショットを参照）。フィルターアイコンをクリック*してフィルターを有効化/無効化することができます。

フィルターに関する以下の点に注意してください：

- オブジェクトと配列のみに適用することができます。
- フィルターの XQuery 式のコンテキストノードは現在のノードです。上のスクリーンショットでは、例えば、フィルターの XQuery 式のコンテキストノードは **`Tracks`** ノードです。

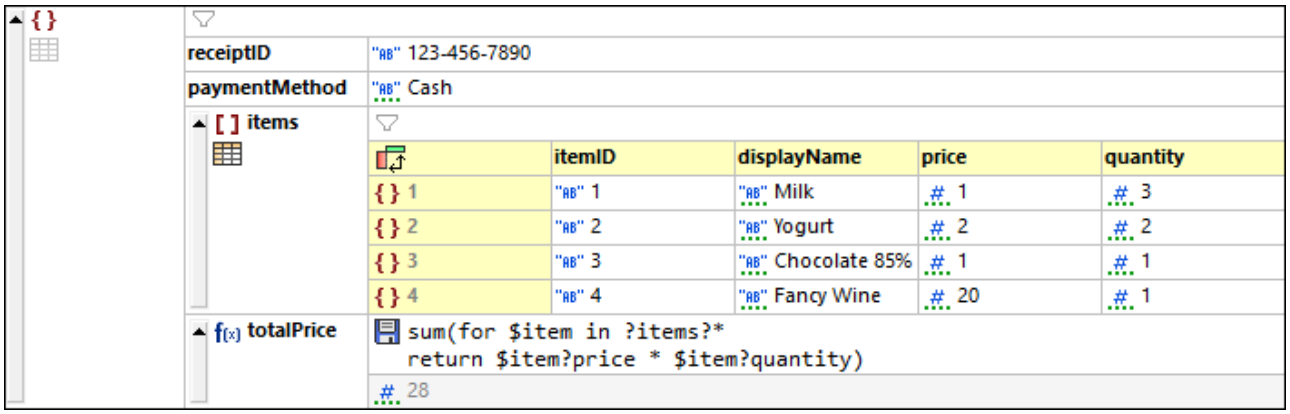
- フィルターはJSON ドキュメント内でJSON コンテンツとしては保管されませんが、次回の使用のために特別なメタデータとして保管することができます。フィルターはドキュメントのメタデータとして保管され、JSON グリッドビュー内でドキュメントが開かれると対応するオブジェクトや配列に適用されます。
- フィルターをネストすることが可能です。ネストされたフィルターは親フィルターのフィルターされたコンテンツに適用することができます。
- 式内に新規のラインを追加するには、「Ctrl+Enter」を押します。

13.5.7　フォーミュラ

フォーミュラはJSON グリッドビュー内の表示のために出力（ノードセットまたは計算）を生成するXQuery 3.1 式です 。下のスクリーンショットでは、例えば、アイテム1 から4 の総額が計算され、出力（28）が個別のラインで表示されます。各フォーミュラは個別に実行され他のフィルターまたはフィルターにより影響されません。

JSON のためのXQuery 式ドキュメントの作成に関する詳細は、JSON のためのXQuery 式 543 を参照してください。

メモ　グリッドビュー内にフィルターとフォーミュラのために式を入力する際に、読みやすいように式を複数行で表示するようにすることが奨励されます。式内に新規のラインを追加するには、「Ctrl+Enter」を押します。



ドキュメントがJSON5 またはJSONC ドキュメントである場合:

- フォーミュラはJSON コメントとしてドキュメントに書き込まれます。
- 任意で、フォーミュラの出力はJSON ドキュメント内の（コメントを持たない）コードとして保管されることができます。これを行うには、フォーミュラのディスクアイコンをクリックします（*下のスクリーンショットを参照*）。JSON コンテンツとしてフォーミュラが保管されない場合、グリッドビュ内で表示されるドキュメントメタデータとして保管されます。JSON5 とJSONC ドキュメント内でのみでディスクアイコンを使用することができ、JSON ドキュメントの他の型では表示されません。

保管されたフォーミュラとフォーミュラの結果を含む上のスクリーンショット内で示されているJSON ドキュメント

*注意点：f*ーミュラは *JSON コメントとして保管されますが、フォーミュラの結果 は ストレートな JSON コードとして保管されます。下の（フォーミュラの結果が保存された）コードはクリックされたフォーミュラのディスクアイコンの結果です*

```
{
        "receiptID": "123-456-7890",
        "paymentMethod": "Cash",
        "items": [
                {
                        "itemID": "1",
                        "displayName": "Milk",
                        "price": 1,
```

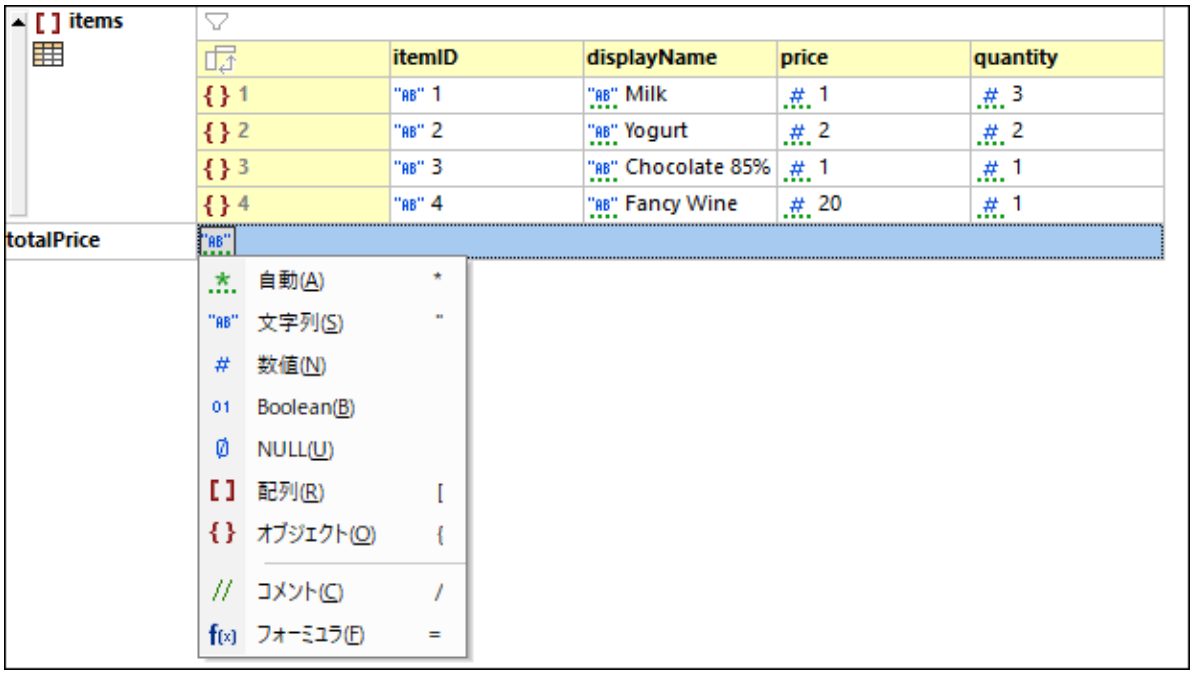
```
                "quantity": 3
            },
            {
                "itemID": "2",
                "displayName": "Yogurt",
                "price": 2,
                "quantity": 2
            },
            {
                "itemID": "3",
                "displayName": "Chocolate 85%",
                "price": 1,
                "quantity": 1
            },
            {
                "itemID": "4",
                "displayName": "Fancy Wine",
                "price": 20,
                "quantity": 1
            }
        ],
        //(:altova_xq_embed:)totalPrice(:altova_xq_key:)sum(for $item in ?items?*
return $item?price * $item?quantity)
        "totalPrice": 28
        //(:altova_xq_end:)
}
```

上のスクリーンショットでは、フォーミュラはシーケンスのメンバーの合計を計算します。メンバーとは **items** 配列内に含まれる各オブジェクトの **price** と **quantity** 値の製品のそれぞれです。各オブジェクトを選択し、**$item** 変数に割り当てるための反復は以下により指定されます: **for $item in ?items?*** 。フォーミュラの親である、したがって **items** ノードの親である、コンテキストノードをメモすることはこの時点で重要です。各製品は **$item** 変数内に現在存在するオブジェクトの **price** と **quantity** 子ノードをルックアップし、これらの値を乗算することにより取得されます。この方法で取得された製品は総額を生成するために合計されたシーケンスのメンバーです。

フォーミュラの作成

フォーミュラを作成するには、以下を行います:

1. フォーミュラに兄弟または子として追加するノードを選択します。右クリックし、兄弟または子を追加します。フォーミュラが（兄弟または子として）表示される個所を決定する場合、フォーミュラの XQuery 式のコンテキストノードはフォーミュラの親ノードになります。例えば、上のスクリーンショット内で表示される XQuery 式内では、コンテキストノードはフォーミュラ（**totalPrice**）とその兄弟、 **items** 配列の親ノードです。上のスクリーンショット内でフォーミュラを作成する場合、兄弟ノードは **items** 配列に追加されます *（下のスクリーンショットを参照）*。
2. フォーミュラに対するノードの型を変更します *（下のスクリーンショットを参照）*。



3. `f(x)` アイコンを含むセルをダブルクリックし、フォーミュラの名前を入力します（*下のスクリーンショットを参照*）。ドキュメントが JSON5 または JSONC ドキュメントの場合、ディスクアイコンが表示されます。このアイコンをクリックして、フォーミュラの出力をドキュメントに保存することができます。

4. デフォルトでは、XQuery 式は文字列 `'XQuery'` ですが、出力は（式の下のセル内に表示されている）文字列 `XQuery` です。XQuery 式を式のセルをダブルクリックして入力し、式を入力し、「Enter」をクリックします。これによりフォーミュラが評価されその結果が表示されます。他のフォーミュラは変更されません。
5. ドキュメント内に複数のフォーミュラが存在する場合メニューコマンド「**JSON | 全てを再評価する**」をクリックして全てのフォーミュラの結果を更新します。ドキュメント内のフォーミュラは（例えば、為替などの）動的に変更するデータをルックアップする場合このコマンドは特に役に立ちます。

フォーミュラに関する以下の点に注意してください：

- フォーミュラの XQuery 式のコンテキストノードはフォーミュラノードの親ノードです。
- フォーミュラはドキュメントの（特別なアプリケーション全体の JSON メタデータファイル内にある）メタデータ内で保管されます。ドキュメントが JSON5 または JSONC の場合、フォーミュラは追加で JSON コンテンツ内のコメントとして保管されます。
- フォーミュラにより生成された出力はフォーミュラの XQuery 式の下のセル内に表示されます。JSON5 と JSONC の場合、XQuery 式の横のディスクアイコンをクリックして出力をドキュメント内に保管することができます。ディスクがクリックされているかに関わらず、フォーミュラの出力は計算されドキュメントのメタデータに保管されます。

	JSON コンテンツに保存されていないフォーミュラ出力。クリックして保存します。*JSON5 と JSONC 内のみ。*
	JSON コンテンツに保存されているフォーミュラ出力。*JSON5 と JSONC 内のみ。*

- 出力が計算でコンテンツに保管されている場合、フォーミュラに割り当てた名前を持つプロパティとして保管されます。例えば、上記の `totalPrice` フォーミュラ内では、出力は以下のようになります：`"totalPrice": 28`.
- 以下の違いに注意してください：JSON5 と JSONC ドキュメント内では、JSON コメントとしてフォーミュラは保存されます。

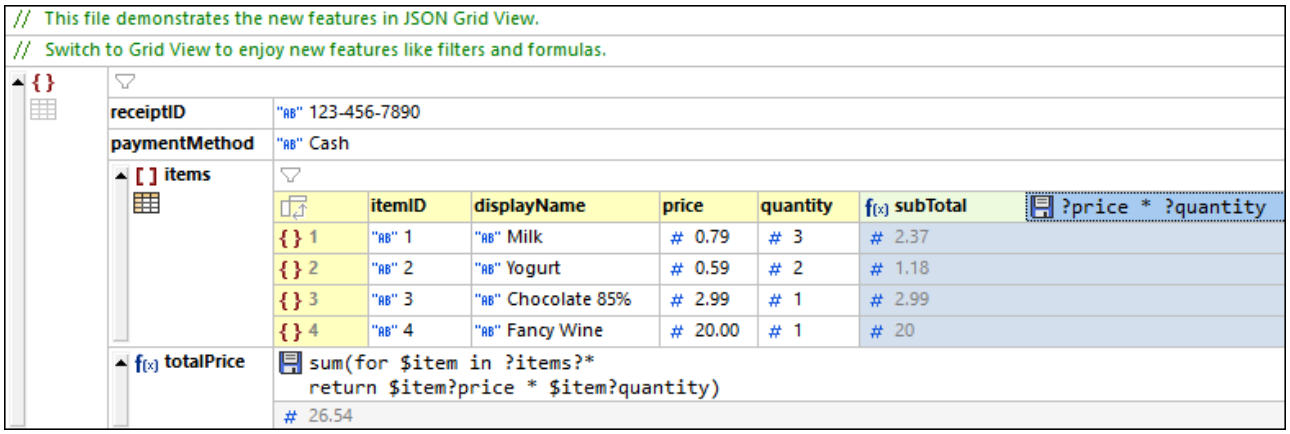
メモ JSON グリッドビュー設定ダイアログ[502] によりフォーミュラがドキュメント内に保管されるかを指定することができます。フォーミュラを保管するオプションはデフォルトで選択されます。

メモ JSON グリッドビュー設定ダイアログ[502] 内で以下を行うことができます：(i) (メタデータファイル内に自動的にフォーミュラを保管するか、または保管しないかなどの) メタデータオプション

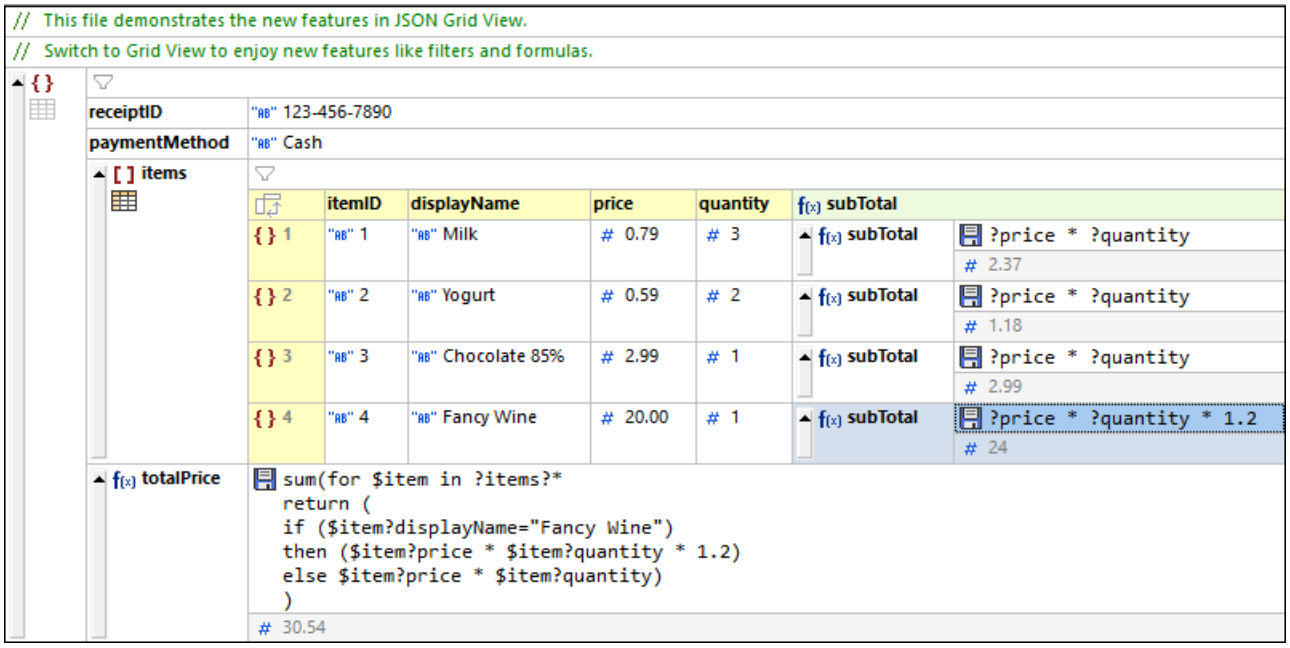
テーブル内のフォーミュラ

(テーブルディスプレイ[488]) 内のテーブル列のすべてのセルに同じフォーミュラが含まれている場合、フォーミュラは列のヘッダー内で一度のみ表示されます (*下のスクリーンショットを参照*)。フォーミュラ計算の*結果*はしかしながら対応するセル内に表示されます。

列ヘッダー内のフォーミュラはグリッドビュー表示です。(テキストビュー内の) JSON ドキュメントコンテンツ 内では、フォーミュラは各テーブル行アイテムのために繰り返されています。



(下のスクリーンショットのハイライトされたセル同様) 単一のフォーミュラが異なる場合でも各フォーミュラは対応するセル内に表示されます。



全てのフォーミュラの再評価
ドキュメント内のすべてのフォーミュラの結果を更新するには、メニューコマンド「JSON | 全てを再評価する」をクリックします。

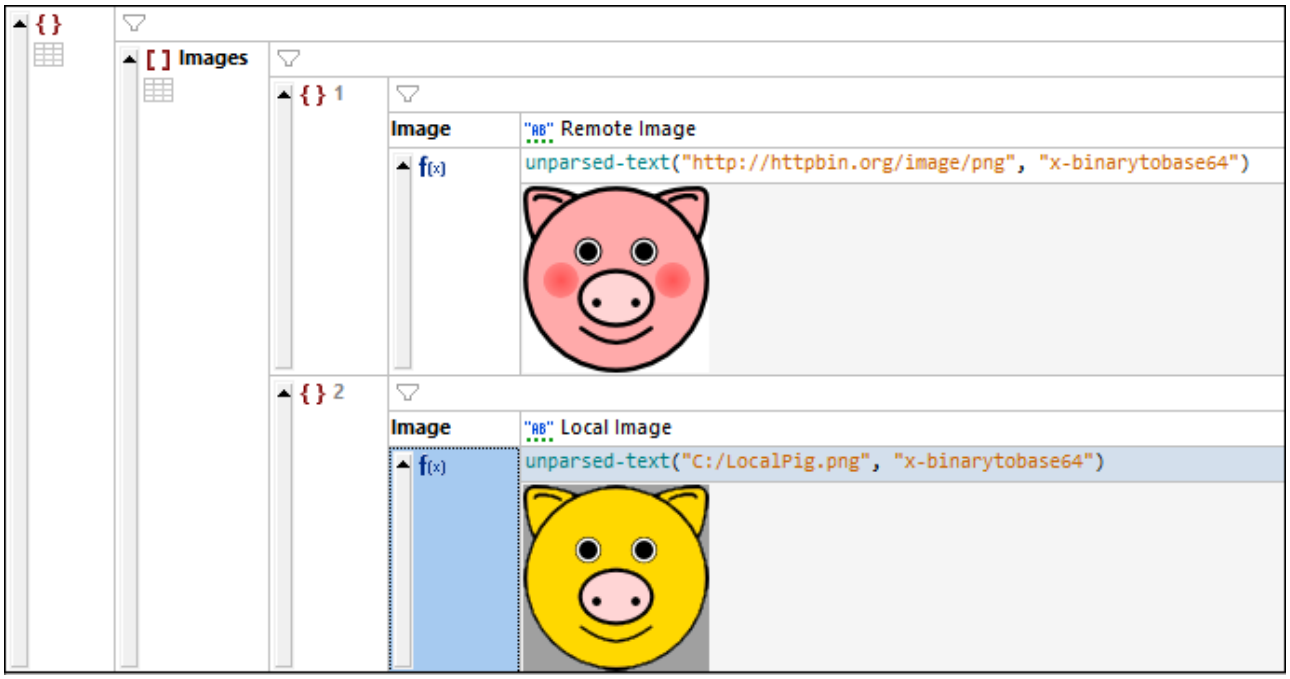
関連したアクションのためのプロシージャ

- 以下のようにテーブルに新規の空の列を追加します: テーブルディスプレイからリスト表示に切り替え、リスト表示内の `key:value` ペアのいずれかをクリックし、新規の `key:value` ペアをアイテムのコンテキストメニューを使用して追加します。テーブルディスプレイに切り替える場合、新規の列が追加または挿入された新規の `key:value` ペアのために作成されます。グリッドビューでこの列を編集することができます。
- フォーミュラがヘッダーに表示されるようにテーブル列のすべてのフォーミュラが同じの場合、および個別のセルのために異なるフォーミュラを作成する場合、リスト表示に切り替え、セルのフォーミュラを編集します。テーブルディスプレイに切り替えると、フォーミュラは(全てのセルのために)個別のセルのために表示されます。
- 式内に新規のラインを追加するには、「Ctrl+Enter」を押します。

13.5.8　イメージ

イメージを JSON グリッドビューで表示することができます (*下のスクリーンショット参照*)。イメージは Base64-エンコード済みイメージに自動的に変換され表示されます。

イメージをロードするには式を使用する際に以下の式を入力します: **`unparsed-text("<Image-URL>", "x-binarytobase64")`**



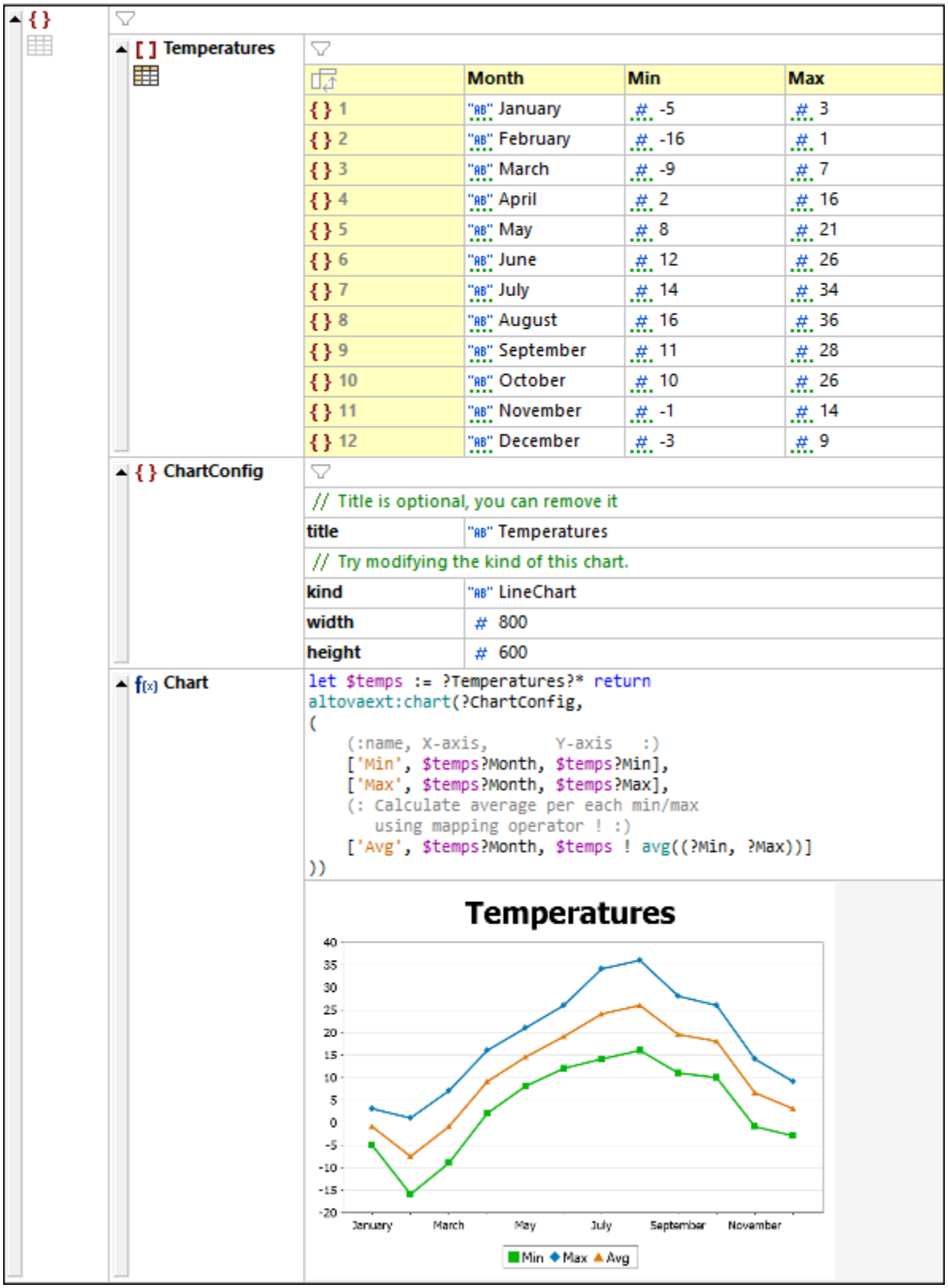
URL を相対的パスを含む以下の書式で提供することができます:

- `http://httpbin.org/image/png`
- `file:///c:/LocalPig.png`
- `C:/LocalPig.png`
- `LocalPig.png`

13.5.9 チャート

Altovaext:ahart という名前のAltova XPath/XQuery 拡張子を使用してJSON グリッドビュー内でチャートを作成することができます（*下のスクリーンショットを参照*）。拡張子は下で説明されています。これに関してはチャート関数のセクション内で他のチャート共に説明されています。下のスクリーンショットで表示されている **altovaext:chart** 拡張子は JSON グリッドビューフォーミュラ 493 内で定義されているXQuery `Let` 式内で使用されています。チャートはフォーミュラの下でイメージとして表示されています。（このチャートサンプルはドキュメント 27 の *Examples* フォルダー内にあるファイル **Chart.jsonc** 内にあります。または、*サンプルプロジェクト* 105 からアクセスすることもできます。）

(C) 2015-2021 Altova GmbH



{ }

[ ] Temperatures

	Month	Min	Max
{ } 1	"AB" January	# -5	# 3
{ } 2	"AB" February	# -16	# 1
{ } 3	"AB" March	# -9	# 7
{ } 4	"AB" April	# 2	# 16
{ } 5	"AB" May	# 8	# 21
{ } 6	"AB" June	# 12	# 26
{ } 7	"AB" July	# 14	# 34
{ } 8	"AB" August	# 16	# 36
{ } 9	"AB" September	# 11	# 28
{ } 10	"AB" October	# 10	# 26
{ } 11	"AB" November	# -1	# 14
{ } 12	"AB" December	# -3	# 9

{ } ChartConfig

```
// Title is optional, you can remove it
title    "AB" Temperatures
// Try modifying the kind of this chart.
kind     "AB" LineChart
width    # 800
height   # 600
```

f(x) Chart

```
let $temps := ?Temperatures?* return
altovaext:chart(?ChartConfig,
(
    (:name, X-axis,       Y-axis   :)
    ['Min', $temps?Month, $temps?Min],
    ['Max', $temps?Month, $temps?Max],
    (: Calculate average per each min/max
       using mapping operator ! :)
    ['Avg', $temps?Month, $temps ! avg((?Min, ?Max))]
))
```

上のスクリーンショット内で表示されているチャートを生成するためのXQuery 式のリスティング

```
let $temps := ?Temperatures?* return
altovaext:chart(?ChartConfig,
(
    (:name, X-axis,       Y-axis   :)
    ['Min', $temps?Month, $temps?Min],
    ['Max', $temps?Month, $temps?Max],
    (: Calculate average per each min/max
       using mapping operator ! :)
    ['Avg', $temps?Month, $temps ! avg((?Min, ?Max))]
))
```

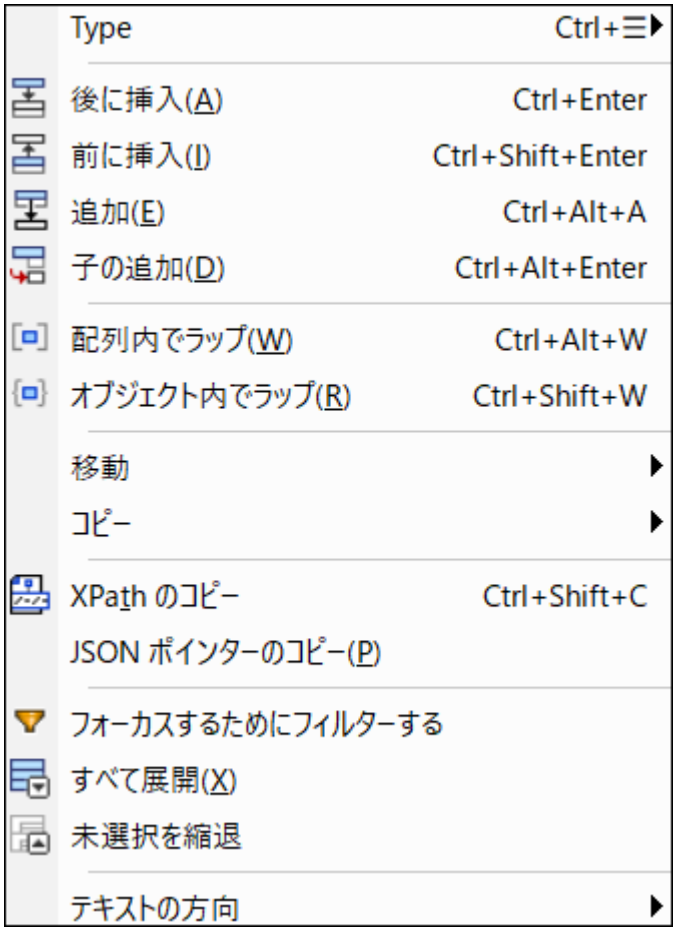
Altova チャート拡張子の使用

- チャート拡張関数 `altovaext:chart` は名前空間プレフィックス `altovaext:` を使用する必要があります。
- `altovaext:chart` は2個の引数を取ります: (I) チャート構成情報と(ii) チャートデータ系列情報。
- チャート構成情報は `altovaext:chart` の最初の引数です。4個のキー—値ペアの順序指定されていない系列です。これらのペアは以下のために存在します: (i) チャートのタイトル(キーは `title`)、(ii) 円グラフ、折れ線グラフなどのチャートの種類(キーは `kind` 使用可能な種類に関してはチャート関数 を参照してください。)、(iii) ピクセルで表示されたチャートの幅(整数のみ。キーは `width` です)。および(iv) ピクセルで表示されたチャートの高さ(整数のみ。キーは `height` です)。上のサンプルのスクリーンショット内では `altovaext:chart` 関数内で参照されている `ChartConfig` オブジェクト内に構成情報が保管されています。
- チャートデータ系列は `altovaext:chart` の2番目の引数です。各データ系列はサイズ3の配列です: (I) 系列の名前 (ii) x-軸の値 (ii) y-軸の値(例えば、上のサンプルのように各系列がラインを表示する場合など) 複数の系列を作成する場合は複数の配列のシーケンスを作成します。上のサンプルには最低、最高、および平均機能のための3個のデータ系列が存在します。XとY 軸のためのデータは `Temperatures` という名前の配列から参照されます。

詳細に関しては、チャート関数のセクションを参照してください。

13.5.10 グリッドビュー内のコンテキストメニュー

JSON グリッドビュー内でコンポーネントを右クリックすると、ドキュメントの表示と編集を変更するためのコマンド表示するコンテキストメニューが表示されます *(下のスクリーンショット)*。コンテキストメニューはマイナスキーを押してアクセスすることができます。コンテキストメニューのコマンドは下で説明されています。



型
型 コマンドにマウスをポイントすると[コンポーネントの型](482) を選択することのできる表示されるサブメニューを表示します。「**Ctrl+キーメニュー**」を使用して型サブメニューにアクセスすることができます。

子を前/後に挿入、子の付加、子の追加
「**挿入**」と「**追加**」コマンドにより同じレベルでアイテムを追加することができます。**後に挿入** と **前に挿入** の場合新規のアイテムがそれぞれ選択されたアイテムの前後に追加されます。**付加** の場合、選択されたアイテムの最後の兄弟として追加されます。「**子の追加**」コマンドは新規のアイテムを子として追加します。1 つまたは複数の子アイテムが既存の場合、新規アイテムの型は最後の子と同じになります。子が存在しない場合、新規のアイテムは空の `key:value` ペアになります。例えば、選択されたアイテムがオブジェクト子を持つ配列の場合、新規の子アイテムは最後の子オブジェクトとして追加されます。

配列内でのラップ、オブジェクト内でのラップ
テーブルの選択されている部分は配列またはオブジェクト内にラップされます。

上/下/左/右への移動
コンポーネントを現在の位置からグリッド内で上下左右に移動しすることが可能になり、対応するコマンドが有効化されます。移動を実行するために対応するコマンド選択してください。

コピー
これらのコマンドは現在の選択項目をクリップボードにコピーするために使用されます。そのコンポーネントに許可されているオプションのみを使用することができます。

コマンド	説明
JSON テキストとしてコピーする	現在の選択範囲はJSON マークアップとしてシリアル化されています。
タブで区切られたテキストとしてコピー	現在のテーブル選択項目がTSV (タブ区切り値)としてシリアル化されます。
イメージとしてコピーする	現在のイメージセルがイメージとしてコピーされます

以下の点に注意してください:

- *イメージとしてコピー*は選択されたイメージのBase64 エンコード文字列をコピーします。(例えば、テーブルの表示内の他のセルなど) Base64-エンコード済み文字列がイメージとして表示されるドキュメントに文字列が張り付けられると文字列は標示されます。それ以外の場合は文字列として解析されます。
- テキストをセルに挿入するには、テキストをコピーしてセルに張り付けます。
- イメージをファイルからインポートするには、[編集 | 挿入 | エンコードされた外部ファイル]768コマンドを使用します。Base64 エンコード文字列としてイメージファイルを挿入し、この文字列をJSON グリッドビューで表示します。インポートのためにサポートされているイメージファイルフォーマットは以下の通りです: PNG、JPEG、BMP、GIF、TIFF。

XPath のコピー
選択されたノードをロケートするドキュメントルートから開始するPath 3.1 ロケーター式をクリップボードにコピーします。

JSON ポインターをコピーする
このコマンドは選択されたノードをロケートするJSON Pointer 式をクリップボードにコピーします。例: **/Artists/1/Albums/1/Tracks**. 詳細に関してはXPointer/JSON-Pointer のコピー768 を参照してください。

フィルター、フォーカスするフィルター
フィルター コマンドは切り替えコマンドです。選択されたオブジェクトにフィルターを追加またはフィルターを無効化にします。このコマンドを使用してフィルターを追加後、フィルター式を入力する必要があることに注意してください。それ以外の場合フィルターは自動的に削除されます。フィルターが追加された後、コマンドをもう一度選択またはフィルターをクリックして無効化することができます。フィルターを削除するにはツールバー内で *全てのフィルターを削除する*をクリックします。

フォーカスするフィルター コマンドは(i) 選択されたノードをロケートするXPath 式を含むルート要素上にフィルターを作成します。そして(ii) 選択されたノード (とその子孫) を直接ルート要素の下で表示するためにグリッドビューをフィルターします。選択されたノードの表示のみに集中する場合この機能は役にたちます。フィルターを削除するには、ルート要素のコンテンツセル内のフィルターアイコンをクリックします。

フルに展開する
このコマンドは選択されているコンポーネントまたはその子孫が折りたたまれている場合に有効化されます。コンポーネントと子孫コンポーネントが展開されます。

選択されていない範囲を折りたたむ
このコマンドは選択されているコンポーネントとその祖先コンポーネント以外のすべてのコンポーネントを折りたたみます。

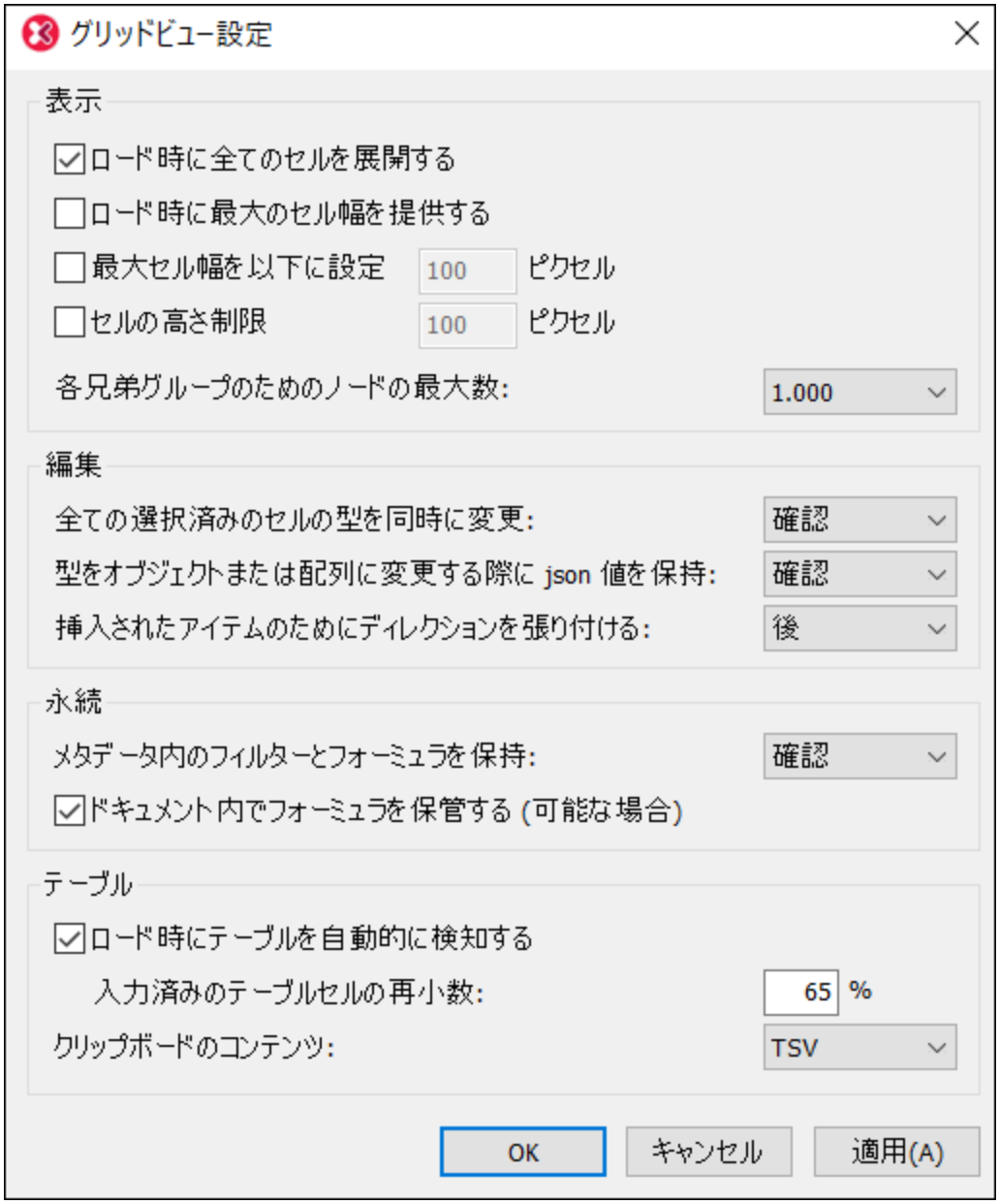
テキストの向き
このコマンドはString 型を含むノードのために使用することができ、読み取りの順序をセルの左側方向または右側方向に切り替えます。この機能はアラブ語とヘブライ語などの言語を使用する際に役に立ちます。

13.5.11 グリッドビューの設定

グリッドビューの設定はグリッドビュー設定ダイアログ (*下のスクリーンショット*)はグリッドビューツールバー477内の設定アイコンを使用してアクセスすることができます。

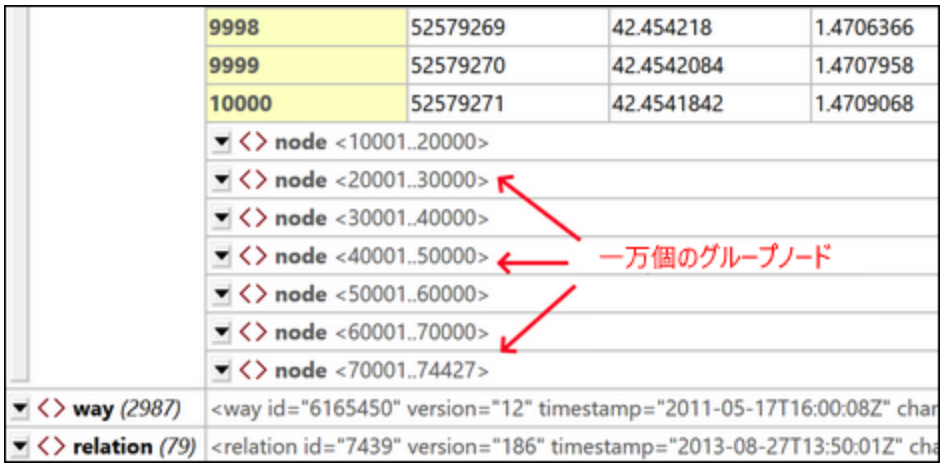
	グリッドビューツールバー477内のグリッドビュー設定アイコン

グリッドビューの設定は以下で説明されています。全てのドキュメント（XML、JSON、DTD）のグリッドビューに適用することができることに注意してください。



表示
表示 セクション内のチェックボックスは自明です。以下は更に説明が必要な場合のメモです。

- ロードする際に全てのセルが展開されていない場合ルートノードとそのすべての子孫は折りたたまれています。ドキュメントを更に深く移動するためには各ノードを展開する必要があります。
- 最適な幅がオンに切り替えられている場合グリッドの幅全体が表示されます。これを達成するには、セルの一部のテキストがラップされます。
- 兄弟ノードは100、1000 または10000 ノードの兄弟グループで表示することができます（*下のスクリーンショットを参照してください*）。多数の兄弟ノードを持つファイル内でスペースの表示を省略する場合



編集

表示 セクション内のチェックボックスは自明です。以下は更に説明が必要な場合のメモです。

- 複数の選択されたセルの型を変更する場合、設定のアクションを継続するかを決定する以下のオプションが与えられます: (ユーザーにより決定される) *常に*、*なし* または *確認* 。
- JSON 型を動的な型からオブジェクトまたは列挙に変更する場合、設定のアクションを継続するかを決定するかを決定する以下のオプションが与えられます: (i) *確認* (動的な型の値が名前を持たない子 `key:value` ペアの値として保持される、または破棄されるかを確認) (ii) *常に* (名前を持たない子 `key:value` ペア内の値を保持する) (iii) *なし* (値を保持する。すなわち値を破棄する)。
- *ディレクションの貼り付け* オプションはクリップボード内の選択範囲が選択されたセルの上または下に張り付けられるかを決定します。

永続

フィルターとフォーミュラを XML と JSON ドキュメントのメタデータ内に保管することができます。フォーミュラは XML ドキュメント内の処理命令として、そして JSON ドキュメント内のコメントとして保管されます。これらの設定はフィルターとフォーミュラの保管の方法を指定します。フォーミュラの場合 *可能な場合* という用語はフォーミュラが実際にすべての JSON ドキュメント (JSON5 と JSONC ドキュメント内でのみ) のコメントとして保存できないことを意味します 。

テーブル

以下の設定を使用することができます:

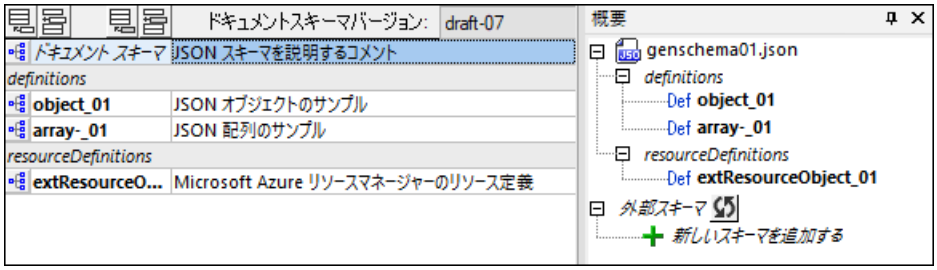
- 設定がロード時に JSON グリッドビューテーブルを自動的に検知する場合、テーブルをテーブルとして検知するために評価する記入されたテーブルセルの最小のパーセントを選択することができます。記入されたテーブルセルの数値がこのレベルを越さない場合、構造はそれぞれが下に表示されている繰り返し要素を持つ通常のグリッドとして表示されます。JSON グリッドビューのみにこの設定は適用されることに注意してください。XML グリッドビュー内では、テーブルセルの記入されている率に関わらずテーブルは自動的に検知されます。
- クリップボードコンテンツがタブにより区切られた値 (TSV) または (ドキュメントの型に従い) XML/JSON として保管するかを選択することが可能です。これはとても役に立つ機能です: クリップボードからテーブルを他のドキュメントに張り付ける場合、この設定により TSV としてまたはマークアップとしてテーブルがコピーされるかを選択できるようになります。(違いを確認するには、テーブルをそれぞれのフォーマットでクリップボードにコピーしテキストエディターにテーブルを張り付けてください。)

13.5.12 JSON スキーマビュー

JSON スキーマビューは、JSON スキーマドキュメントを閲覧し編集するために使用することができます。JSON スキーマビューウィンドウの主な部分は以下のとおりです:

- 定義概要グリッド[507] とデザインビュー[513] 間を切り替えるメインウィンドウ
- 3 つの入力ヘルパーウィンドウ[509] (メインウィンドウの右側にデフォルトで表示されます): 概要、詳細、制約
- メッセージウィンドウ(メインウィンドウの下にデフォルトで表示されます)
- 情報ウィンドウ[537] (アプリケーションウィンドウの左側にデフォルトで表示されます)

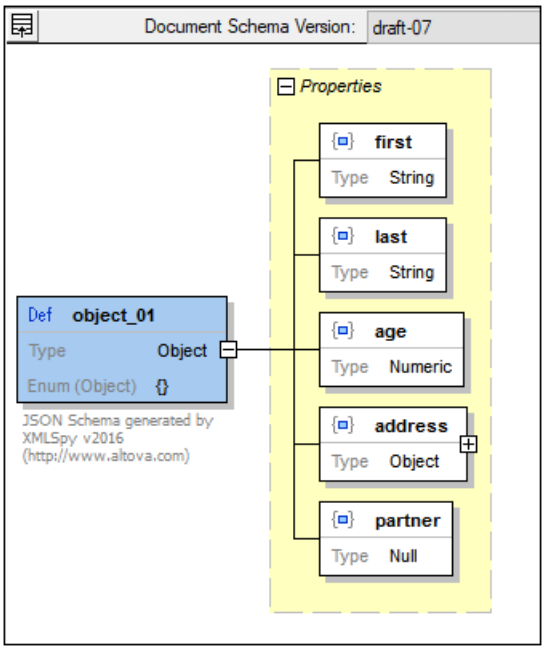
下のスクリーンショットは、メインウィンドウおよび概要入力ヘルパーを表示しています。



メインウィンドウ

メインウィンドウは、定義概要グリッド[507] (*上のスクリーンショットに表示*) とデザインビュー[513] (*下のスクリーンショット*) を切り替えます。定義概要グリッドは、現在のドキュメントの(*"ドキュメントスキーマ"として リストされている*) メインスキーマを表示し、またスキーマに追加する定義を表示します。(定義は、JSON データ構造の詳細です。上のスクリーンショットでは、`object_01` および `array_01` がそれぞれオブジェクトと配列の定義です。) 定義は、概要入力ヘルパーでもリストされています(*上のスクリーンショット参照*)。

定義概要グリッドは、JSON スキーマのハイレベルな表示を行いますが、概要内にリストされている定義の内容を表示しません。デザインビュー内の定義を確認して編集するには(*下のスクリーンショット*)、定義のアイコン(*上のスクリーンショット参照*) をクリックするか、または概要入力ヘルパー内の定義をダブルクリックします (*上のスクリーンショット参照*)。



デザインビューから定義概要グリッドに切り替えるには、デザインビューの左上にある「**定義グリッドに切り替え**」アイコンをクリックします（*上のスクリーンショット参照*）。デザインビューを構成するには、メニューコマンド スキーマデザイン | 構成ビュー 532 をクリックします。

入力ヘルパー

スキーマビューの2つのモード（定義概要グリッド とデザインビュー）には、3 つの入力ヘルパーがあります: 概要、詳細、制約。これらの入力ヘルパーは以下のためのメカニズムが搭載されています: (i) スキーマとその定義に関する情報を表示 (ii) 定義に関連する情報と値の入力。詳細に関しては、入力ヘルパー: 概要、詳細、制約 509 を参照してください。

13.5.12.1 JSON スキーマバージョン

JSON スキーマは、JSON 構文で書かれており、`.json` 拡張子を持つファイルに通常保存されています。以下の値を持つトップレベルのオブジェクトの最初のキーワードである **`$schema`** キーワードによりJSON スキーマとして表示されています: **`"http://json-schema.org/draft-N/schema#"`** または **`"http://json-schema.org/schema#"`**.

```
{
   "$schema": "http://json-schema.org/draft-07/schema#",
   ...
}
```

or

```
{
   "$schema": "http://json-schema.org/schema#",
   ...
}
```

上記の最初のサンプルでは、スキーマバージョンは **`draft-04`**、**`draft-06`**、などの書式を持つバージョン名と共に明示的に名前が付けられています。上の2番目のサンプルの通り、バージョン名が省略されている場合、これは使用されるスキーマバージョンは最新のバージョンであることを示しています（現在は **`<%JSON-SCHEMA-LATEST%>`**）。

JSON スキーマビュー内で、メインウィンドウの上のバー内のコンボボックス内でバージョンを変更することができます（*下のスクリーンショットを参照してください*）。

他の古いバージョンで定義されていない新規スキーマバージョンの機能

新しいスキーマバージョンの機能を使用し、この機能をサポートしない古いバージョンに切り替える場合、以下が発生します:

- 機能を削除/変換、または、機能を保管するかを問うメッセージが表示されます。
- 保管する場合、コンポーネントまたは詳細に対応する新規機能がオレンジ色のテキスト色で表示されます。 例えば、値が `const` キーワードに設定されており（`draft-06` の新規機能）`draft-04` にスキーマバージョンを変更すると、`const` キーワードの値がオレンジ色で表示されます。

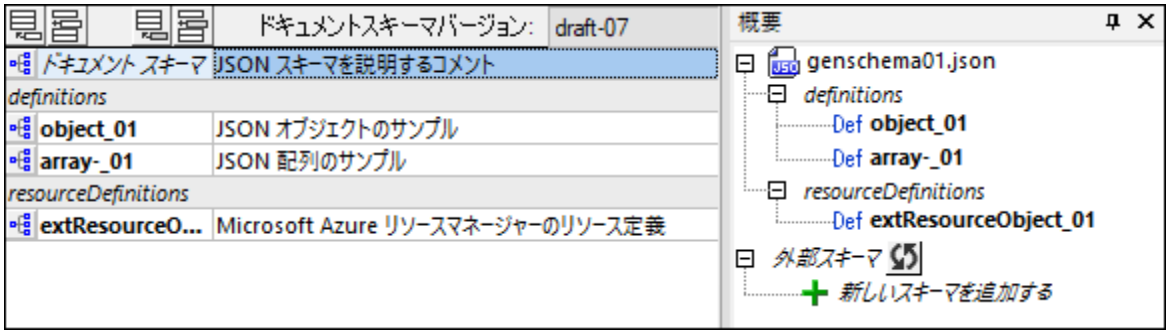
JSON スキーマバージョン

特に各バージョンに関する追加情報についてのJSON スキーマ仕様の詳細に関しては、下のリンクを参照してください:

JSON スキーマ仕様: http://json-schema.org/specification.html
Draft-06 リリースノート: http://json-schema.org/draft-06/json-schema-release-notes.html
Draft-07 リリースノート: http://json-schema.org/draft-07/json-schema-release-notes.html

13.5.12.2 グローバル定義の追加

メインウィンドウ内の定義概要グリッド 505（*下のスクリーンショット*）は、スキーマのグローバル定義を表示します。これらのグローバル定義は、以下のとおりです: (i) **メインドキュメントスキーマ** 定義、(ii) オブジェクト、配列、文字列などの、JSON スキーマ型である **グローバルな JSON 型**の定義、(iii) **外部、または、カスタムに定義された JSON 型** の定義。`resourceDefinitions` という名前のコンテナー内で発生する定義のみを使用することができます。これは、JSON 定義のためのMicrosoft Azure Resource Manager により使用されているコンテナーです。グリッドツールバーの **定義セクションの追加** または **定義セクションの挿入** アイコンを使用して、新しいリソース定義を追加します（*下のスクリーンショットを参照してください*）。

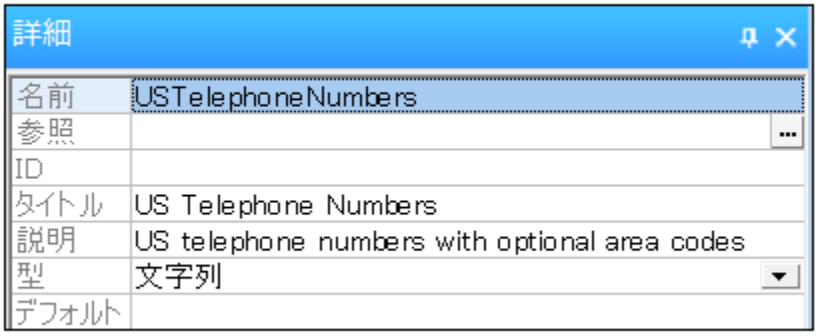


JSON 型 468 とグローバルに定義することは、同じスキーマおよび他のスキーマ内でその型を使用する際にとても便利です。例えば、1 つの JSON スキーマ内で JSON 文字列の型を米国の電話番号で使用すると仮定します。 同じスキーマと他の JSON スキーマからもこの定義を参照することができます。

定義と関連したアクションの追加

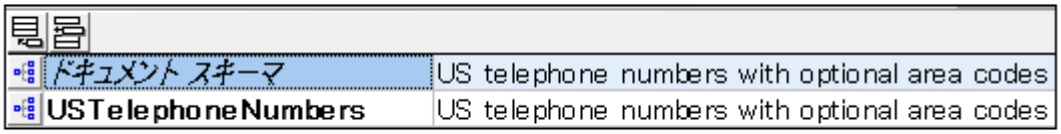
定義概要グリッド内で定義を追加および編集するために次のアクションを使用することができます。

- *定義を追加する:* 定義概要グリッドの左上の「**名前を付けられたスキーマ 定義を追加**」または「**名前を付けられたスキーマ 定義を挿入**」をクリックします（*上のスクリーンショットを参照*）。新しい空の定義がグリッド内に作成されます。新規の定義は、概要入力ペインにも *Def* としてリストされます（*上のスクリーンショットを参照*）。
- *定義の型の変更:* `Any` の型の新しい定義が作成されます。詳細入力ペインで型を変更することができます（*下のスクリーンショットを参照。型が 'String' である箇所*）または、デザインビュー内の定義を編集。



- *定義の名前の変更:* 名前をダブルクリックして編集します。 または、詳細入力ペイン内の*名前* フィールドを編集します。
- *定義の説明の入力:* 詳細入力ペイン内の*詳細* フィールドを編集します。定義の名前の横の定義概要グリッド内に詳細が表示されます（*下のスクリーンショットを参照*）。詳細を編集するために定義概要グリッドの詳細フィールドをダブルクリックすることもできます。

(C) 2015-2021 Altova GmbH



- *定義の参照:* 詳細に関しては、概要入力ヘルパー[509] の説明とトピックグローバルとローカル定義[512]を参照してください。
- *定義の編集:* 定義概要グリッド内の定義アイコンをクリック、または概要入力ヘルパーの定義をダブルクリックします。これにより定義を編集することのできるデザインビュー内の定義が開かれます。

13.5.12.3 入力ヘルパー: 概要、詳細、制約

JSON スキーマビュー入力ヘルパーは、デフォルトで、アプリケーションウィンドウの右側に表示されます。メインウィンドウの両方のモード[505]で使用することができます：(i) 定義概要グリッド、(ii) デザインビュー。入力ヘルパーウィンドウのタイトルバーをドラッグしてスクリーンの他の場所に移動することができます。入力ヘルパーのタイトルバーを代わりにダブルクリックして、入力ヘルパーをドッキングまたはドッキング解除することができます。詳細に関しては、入力ヘルパー[107]のセクションを参照してください。

概要入力ヘルパー

概要入力ヘルパー（*下のスクリーンショット*）は、現在のスキーマの定義および現在のスキーマのすべてのグローバル定義をリストします。定義をダブルクリックすることにより、デザインビュー[513] 内でその定義を開き編集することができます。ことができます。外部スキーマからの定義を使用するには、最初に外部スキーマを追加して、定義を再利用します。

外部スキーマの追加

概要入力ヘルパー内の「**新しいスキーマの追加**」アイコンクリックして、追加するスキーマを参照し、外部スキーマを追加します。スキーマが追加されると、スキーマが追加されると、その定義は概要入力ヘルパーに表示されます。下のスクリーンショットは、例えば、スキーマ `TelNumbers.json` が追加されたこと、また、このスキーマには、`USTelephoneNumbers` という名前の1つの定義があること表示しています。外部スキーマは任意の数追加することができます。

外部定義の再利用

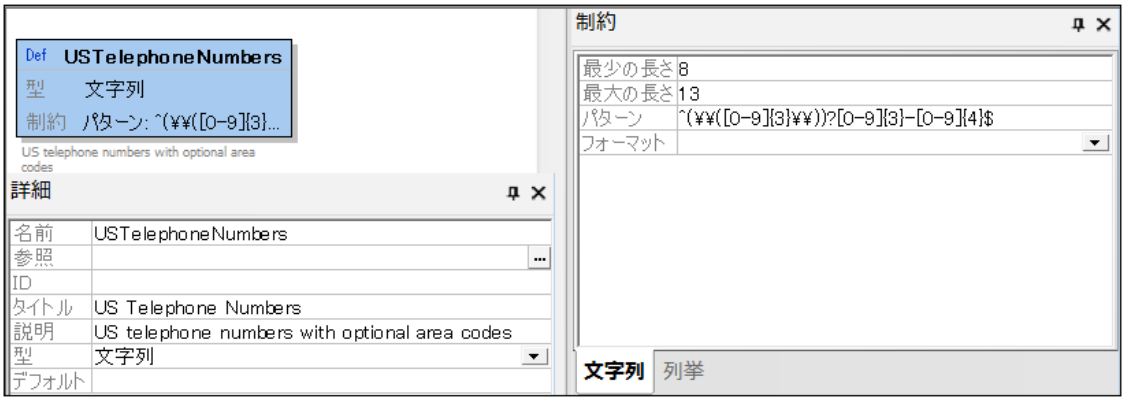
外部スキーマが追加された後、インポートされるスキーマの定義内で再利用することができます。定義の1つが他の定義を（参照して）使用する場合、その定義のプロパティを取ります。参照は以下の2つの方法で行うことができます:

- *デザインビュー内:* 概要入力ヘルパーから定義を定義が使用される箇所にドラッグします。
- *定義概要グリッド または デザインビュー内:* 他の定義を再利用する定義の詳細入力ヘルパーの参照フィールドを使用して。この点に関しては、詳細入力ヘルパー510で説明されています。

メモ T概要ウィンドウ内の外部スキーマの横の「**更新**」アイコンは、追加されたすべての外部スキーマを更新します。追加された外部スキーマから定義が再利用されない場合、そのスキーマはリストが更新されると削除されます。

詳細入力ヘルパー

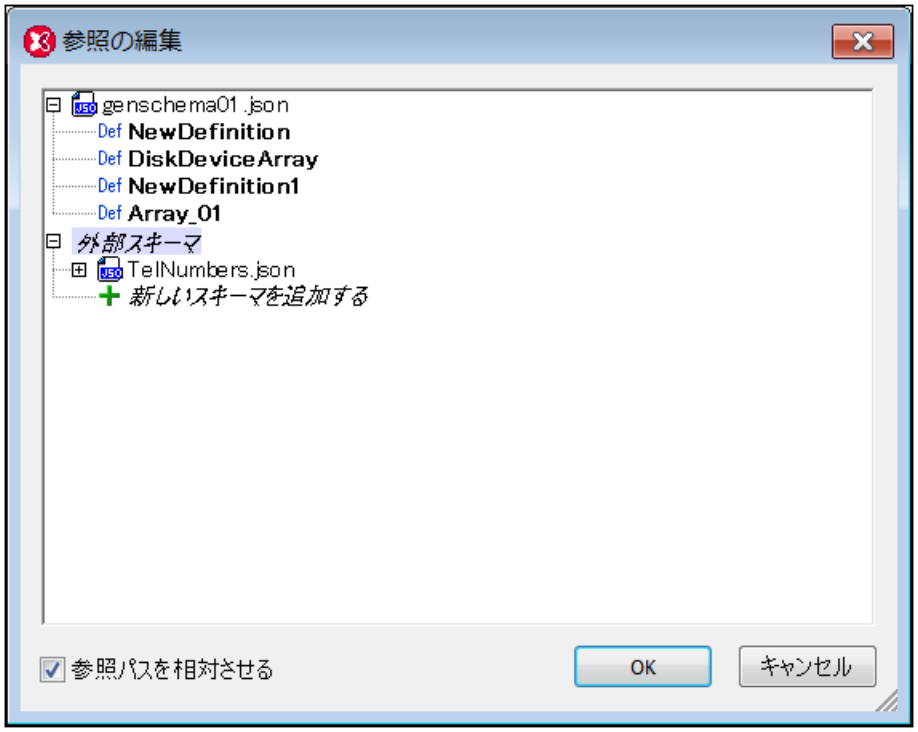
定義のプロパティは、メインウィンドウのいずれかのモード内で定義が選択されると、詳細と制約ウィンドウ内に入力することができます: 概要グリッド、または、デザインビューの定義505. 下のスクリーンショットは、詳細と制約入力ヘルパーと共にデザインビュー内の `USTelephoneNumbers` の定義を表示しています。2つの入力ヘルパー内の情報はデザインビュー513の定義（青）ボックスに表示されていることに注意してください。 2つの入力ヘルパーに設定することのできるプロパティは下にリストされています。



詳細入力ヘルパーに次の詳細を入力することができます:

- *名前:* 定義の名前.
- *参照:* 定義で他の定義を再利用する場合、参照フィールドの「**追加ダイアログ**」ボタンをクリックします。これにより使用可能なすべての（現在のスキーマおよび外部スキーマからの）定義をリストする参照の編集ダイアログが表示されます（*下のスクリーンショット*）。再利用する定義を選択して、相対パスを使用する場合、相対パスのオプションを選択して、「**OK**」をクリックします。

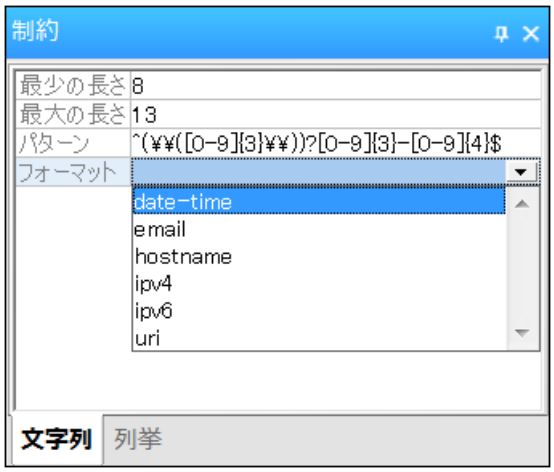
(C) 2015-2021 Altova GmbH



- *型:* コンボボックスのドロップダウンリストから定義のデータ型を選択します。前の定義に戻るには、元に戻す (Ctrl-Z) を押します。これらの型は JSON 型 465 と型セレクタ (Any、Multiple、など) 527 で説明されています。
- *ID:* これはオプションのキーワードで、(親スキーマ内のサブスキーマとして考えられる) 現在の定義のソリューションスコープを変更するために使用されます。ID 値は、URI である文字列である必要があります。Altova JSON バリデーターは、正規の逆参照のみを使用することに注意してください。詳細に関しては、JSON 仕様を参照してください。
- *タイトル、説明:* これらの2 つのキーワードはエンドユーザーにより読み取られることのできる詳細の目的のために使用されます。
- *コメント (draft-07 の新規機能):* エンドユーザーのための詳細に対して、スキーマの管理者のためのメモを想定しています。
- *制約 (draft-06 の新規機能):* 1 つの値の列挙のような定数です。
- *デフォルト:* 定義のデフォルトの値。
- *読み取り専用、書き込み専用 (draft-07 の新規機能):* これらは、それぞれ、読み取りと書き込み専用のフィールドを示しています。書き込み専用のフィールドはパスワードフィールドです。

制約入力ヘルパー

定義の制約は、型により異なります。各型の制約は、以下に説明されています (次も参照: アトミック型 526)。



型が下のリストに表示されない場合、制約を定義することはできません。しかしながら、すべての型に対して列挙を定義することはできます:

- *文字列:* 文字列の長さと文字列のパターン。パターンは正規表現を使用して指定することができます。フォーマットフィールドから仕様内で定義されているフォーマットの1つを選択することができます(*上のスクリーンショットを参照*)。*フォーマット フィールド*内で、,仕様内で定義されている文字列のフォーマット の一つを選択することができます(*draft-04 内で使用することのできるフォーマットを表示する上のスクリーンショットを参照してください*)。追加フォーマットは以降のバージョンで定義されています。(draft-07 での新規機能である) *コンテンツメディア型とコンテンツエンコード* はJSON 文字列でエンコードされた非-JSON データのメディア型とエンコードを選択します。
- *数値:*許可されている数値の範囲
- *配列:*許可されている配列内のアイテムの数
- *オブジェクト:*許可されているプロパティの数

すべての型のための制約入力ヘルパーには列挙タブがあります。その定義の型で許可されているアイテムのリストを指定することができます。更に、*禁止された* 以外のすべての型のためにサンプルタブを使用することができます (draft-06 内では新規) 。これは検証効果の無いサンプルの配列です。`default` の値は、このキーワードの下で繰り返しの無いサンプルとして使用することができます。

13.5.12.4 グローバルとローカル定義

JSON スキーマ定義 グローバルまたはローカルで設定することができます。

- **グローバル定義** は、定義を追加し、プロパティを指定することにより[507]メインウィンドウの定義概要グリッド[505]内で作成されます。グローバル定義は、同じスキーマまたは他のスキーマ内の定義により参照されることができます。これによりプロジェクト全体での定義の再利用が有効化されます。現在のスキーマのすべてのグローバル定義は、スキーマの定義概要グリッド[505]に表示されています。他のスキーマからのグローバル定義は、(概要入力ヘルパー内の)外部スキーマの追加[509] により再利用のために使用可能になります。
- **ローカル定義** は、グローバル定義に子孫 または 兄弟定義 (プロパティまたはアイテム) を追加することにより、グローバル定義内で作成されます。

グローバル定義の再利用

グローバル定義を再利用するには、以下の1つを行います:

- デザインビューで、グローバル定義を概要入力ヘルパーからドラッグして使用する定義の上にドロップします。

- デザインビュー内の定義概要グリッドから、再利用する定義を選択します。詳細入力ヘルパーの参照フィールド内で、再利用するグローバル定義を選択します。詳細に関しては、詳細入力ヘルパー[510] を参照してください。

メモ 同じスキーマ内の他の定義で参照された後に、グローバル定義の名前を変更する場合、参照内で名前も変更されます。他のスキーマからの参照は、しかしながら、名前変更を反映するために手動で編集される必要があります。

ローカル定義 を グローバル定義に変換する

ローカル定義を変換するには、デザインビュー内で定義を右クリックし、「**グローバルにする**」を選択します。グローバル定義が作成され、ローカル定義上に参照が作成されます。グローバル定義の名前は、自動的に生成されるため、変更するとその変更はローカル定義の参照にパスされます。、

グローバル定義の参照をローカル定義に変更する

グローバル定義への参照は、ローカルおよびグローバル定義上に存在することができます。参照を削除して、プロパティをローカルにするには、デザインビュー内の(ローカルまたはグローバル)定義を右クリックして、「**ローカルにする**」を選択します。グローバル定義のプロパティは、定義上にローカルで作成されます。

13.5.12.5 デザインビュー

デザインビューでは、構造および許可されているグローバル定義の値を指定することができます。定義は次の GUI コンポーネント または メカニズムを使用して指定することができます:

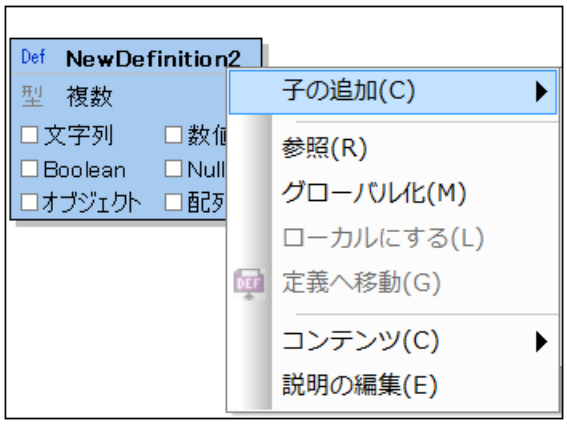
- 詳細入力ヘルパー[510] (定義概要グリッド内で使用することもできます)
- 入力ヘルパーの制約[511] (定義概要グリッド内で使用することもできます)
- 定義のコンテキストメニュー(メインウィンドウの定義ボックスを右クリックしてアクセスすることができます)

詳細と入力ヘルパーの制約により指定することのできる定義は、入力ヘルパー: 概要、詳細、制約[509] のセクションで説明されています。これらのプロパティの一部は、メインウィンドウの定義ボックスで指定することもできます。このセクションでは、次の 3 つのセクションで、メインウィンドウで使用することのできるグラフィカルなメカニズムについて説明を行います。

メモ 誤った操作および希望しない変更を元に戻すには、「Ctrl+Z」を押してください。

コンテキストメニュー

(*下のスクリーンショットの青のボックス*) のコンテキストメニューにより、定義の構造のデザイン、およびプロパティの編集を有効化することができます。

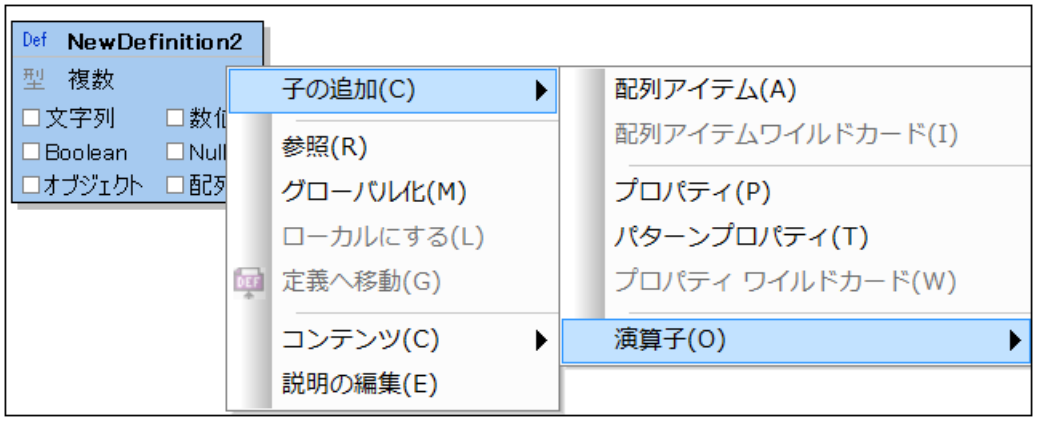


以下のコマンドを使用することができます:

- **子の追加**: 定義の型により追加することのできる子が決定されます（下の子の追加: 構造の作成[514] を参照してください）。
- **参照**: 定義がグローバル定義を参照することを有効化し、そのグローバル定義のプロパティを取ります。コマンドが開く参照の編集ダイアログは、詳細入力ヘルパー[510] を使用してアクセスすることのできるダイアログと同じです。詳細は入力ヘルパー: 概要、詳細、制約[510]を参照してください。
- **グローバルに設定**: このコマンドは、ローカル定義[512]が定義の場合に有効化されます。現在選択されている定義をグローバル定義[512]とし、現在の選択にそのグローバル定義の参照を追加します。 I
- **ローカルに設定**: このコマンドは、グローバル定義[512]が定義の場合に有効化されます。現在選択されているローカル定義[512]を元のグローバル定義への参照を作成して変換します。
- **定義に移動**: 選択された定義がグローバル定義を参照する定義に含まれる場合、このコマンドは有効化されます。クリックするとグローバル定義に移動します。
- **コンテンツ**: **コンテンツ** コマンドは、選択された定義のコンテンツを切り取り、コピーおよびリセットするを含むサブメニューを表示します。
- **詳細の編集**: 編集する定義の詳細フィールドを有効化します。

子の追加: 構造の作成

定義の構造は、複数のレベルの子孫を追加することにより作成されます。これらのレベルは、コンテキストメニューの**子の追加** コマンドにより作成されます。定義に追加する子はその型により異なります:



- *オブジェクト*[515]: プロパティと演算子を取ります

- *配列*[524]: 配列 アイテムと演算子を取ります
- *アトミック型（文字列、数値、boolean、null）*[526]: 演算子を取ります
- *Any*[527]: プロパティ、配列 アイテム、と演算子を取ります
- *Multiple*[527]: 含まれる型によります。選択された方のための許可された子の和集合を取ります。
- *演算子*[530]: 論理演算子を構造を決定するために使用することを有効化します

各型のために作成される構造は、上のリストにリンクされているセクションで説明されています。

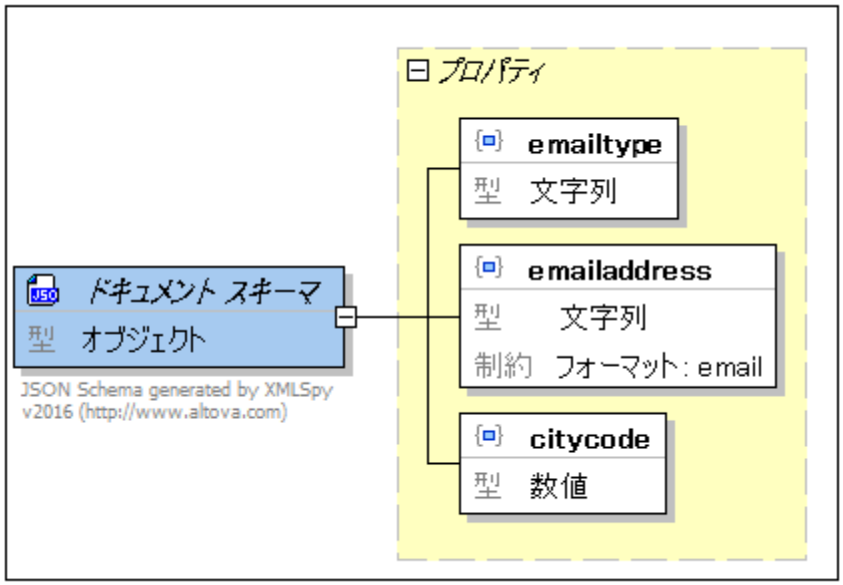
13.5.12.6 オブジェクトとプロパティ

オブジェクトは、中かっこで区切られており、値にキーを以下のようにマップします: `"MyKey": Value`。キーは常に文字列である必要があり、そのために引用符で囲まれている必要があります。値は、JSON データ型[465] 任意のであることができます。各 `key:value` ペアはオブジェクトのプロパティとして知られています（*下のスクリーンショット参照*）。

以下は、3 つのプロパティを持つインスタンス化されたオブジェクトの例です:

```
{
  "emailtype": "home",
  "emailaddress": "contact01.home@altova.com",
  "citycode": 22
}
```

オブジェクトのためのスキーマは、デザインビューで以下のように表示されます。



以下の点に注意してください:

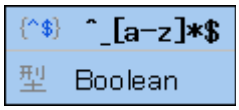
- それぞれのプロパティは、インスタンス内に存在する必要があります。これは、プロパティの境界により示されています。プロパティがオプションである場合、境界は点線で表示されています。プロパティが必要であるかまたはオプションであるかをプロパティのコンテキストメニューまたは詳細入力ヘルパーで設定することができます。
- プロパティがインスタンス内で発生する順序は、スキーマ内で定義することはできません。これは、スキーマ内で定義されるプロパティの順序は、無関係であるからです。

- かっこ内の青い角のシンボルは、(他のシンボルにより示されているプロパティワイルドカード または プロパティワイルドカードに対して) プロパティを示しています。下を参照してください。516.
- プロパティの型は、ダイアグラム内の型をダブルクリックして編集することができ、表示されるドロップダウンリストからオプションを選択することができます。 または、型は詳細入力ヘルパー内で選択することができます。
- `emailaddress` プロパティの制約の値は、入力ヘルパーの制約で定義することができます。

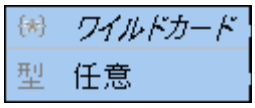
プロパティ、パターンプロパティ、プロパティワイルドカード、およびプロパティ名スキーマ

オブジェクトは、プロパティ、パターンプロパティ、プロパティワイルドカード、およびプロパティ名スキーマ(draft-07 の新規機能506). を持つことができます。これらは、オブジェクト コンテキストメニューを使用して追加することができます: (i) オブジェクト、(ii) 黄色いプロパティボックス(ボックスのプロパティタイトルを右クリック)、(iii) それぞれのプロパティ。プロパティは上で説明されています。ここでは、パターンプロパティとプロパティワイルドカードについて説明をおこないます。

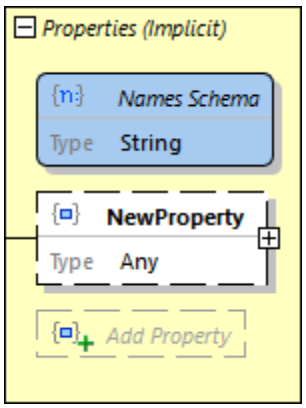
パターンプロパティ (*下のスクリーンショット*) は、プロパティの名前を正規表現として定義します。下のスクリーンショットでは、例えば、正規表現は以下を指定します: (i) プロパティがアンダースコアで始まる名前を有する、そして、(ii) ブール値を値として有する。ワイルドカードのための制約条件はありません。パターンプロパティは任意の数追加することができます。パターンプロパティのためのアイコンに注意してください。



プロパティワイルドカード (*下のスクリーンショット*) は、オブジェクトのプロパティセットの他のプロパティに加え発生するプロパティを指定します。ワイルドカードはしかしながら、これらの発生のための型を定義します。下のスクリーンショットは、数値以外の値を持つ名前を持つプロパティを定義するプロパティワイルドカードを表示しています。各オブジェクトにプロパティワイルドカードは1つのみ存在することができます。ワイルドカードが*Any* 型、に設定されている場合は、入力ヘルパーの制約内のそれぞれの方の制約を設定することができます。プロパティワイルドカードのためのアイコンに注意してください。



プロパティ名スキーマ (*下のスクリーンショット*) はそのメンバーのプロパティの名前を制約します(これは draft-07 の新規機能506です。) 例えば、下のスクリーンショットでは、プロパティの名前は文字列である必要があることがわかります。更に、制約入力ヘルパーを使用してプロパティ名のために更に制約を指定することができます: 例えば、プロパティの名前はプロパティ名は特定の文字の長さにおさまる、または、特定のパターンを持ちます。



メモ プロパティワイルドカード または プロパティワイルドカードの最小および最多の発生設定はありません。更に深い理解のためにプロパティの検証517を参照してください。

プロパティの検証方法

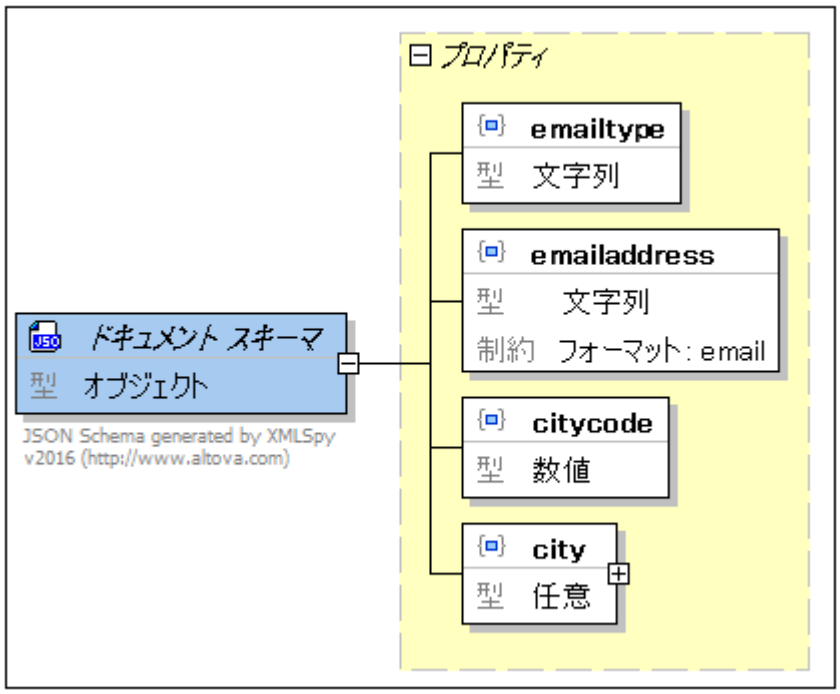
プロパティがインスタンスに遭遇した場合、以下の方法で検証することができます:

1. スキーマ内のプロパティの名前は、オブジェクトのすべての名前付けられたプロパティに対してチェックされます。
2. 一致が検出されない場合、名前はオブジェクトのプロパティセット内のすべてのパターンプロパティに対してチェックされます。
3. まだ一致が検出されない場合、存在する場合はワイルドカードが呼び出されます。
4. まだ名前との一致が検出されない場合、有効性エラーが報告されます。プロパティまたはパターンプロパティに一致する名前、またはワイルドカードが存在する場合、値がプロパティ定義に対応するに値に対してチェックされます。
5. インスタンス値が対応するプロパティ定義の型と制約に一致する場合、プロパティは有効です。それ以外の場合は無効です。

サンプル

下のスクリーンショットは、以下の条件を持つオブジェクトを定義しています:

- `emailtype`、`emailaddress`、および `citycode` という名前を持つ 3 つのプロパティが必要です。
- アンダースコアで始まる名前を持ち、ブール値を持つ 1 つまたは複数のプロパティを持つことができます(下のスクリーンショットのパターンプロパティを参照)。
- 任意の名前と値を持つ 1 つまたは複数の追加プロパティを持つことができます。



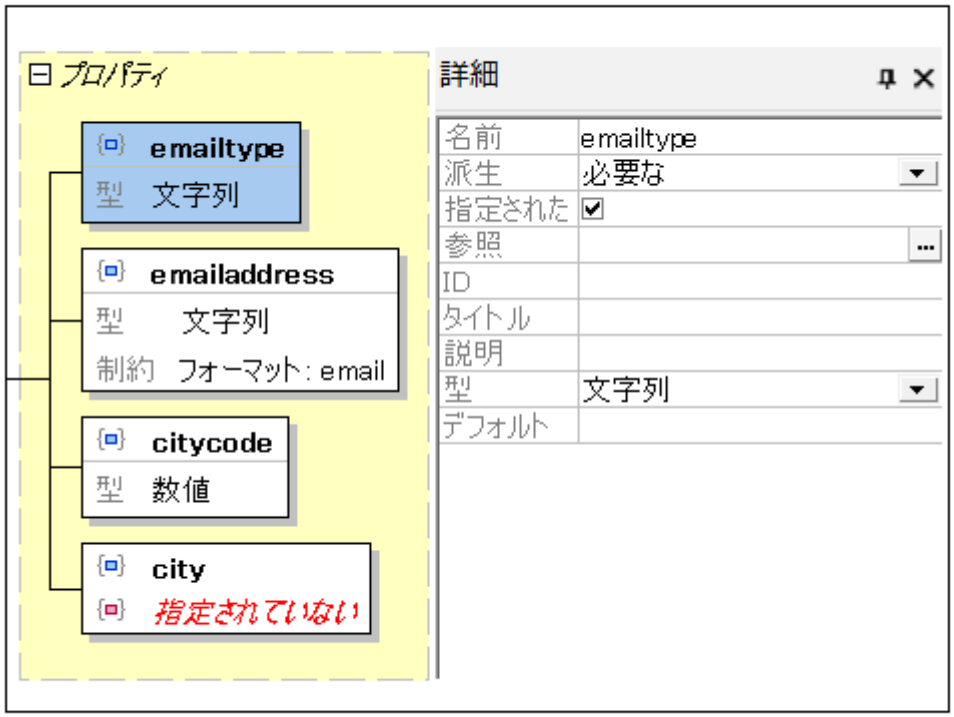
13.5.12.7 指定されていないプロパティ

下のコードのリストでは、`required` キーワードは、4 つのプロパティがこのオブジェクトのために必要なことを指定しています。しかしながら、4 つのプロパティの 3 つは定義されています。4 番目のプロパティ `city` は定義されていません。定義されているプロパティは指定されなければならず、定義されていないプロパティは、指定されていません。リストの下のスクリーンショットを参照してください。

コードのリスト: 指定されているおよび指定されていないプロパティ

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "description": "JSON Schema generated by XMLSpy v2016 (http://www.altova.com)",
    "type": "object",
    "properties": {
        "emailtype": {
            "type": "string"
        },
        "emailaddress": {
            "type": "string",
            "format": "email"
        },
        "citycode": {
            "type": "number"
        }
    },
    "required": [
        "emailtype",
        "emailaddress",
        "citycode",
        "city"
    ],
    "additionalproperties": false
}
```


(C) 2015-2021 Altova GmbH


プロパティ
emailtype
型 文字列
emailaddress
型 文字列
制約 フォーマット: email
citycode
型 数値
city
指定されていない
詳細
名前 emailtype
派生 必要な
指定された
参照
ID
タイトル
説明
型 文字列
デフォルト

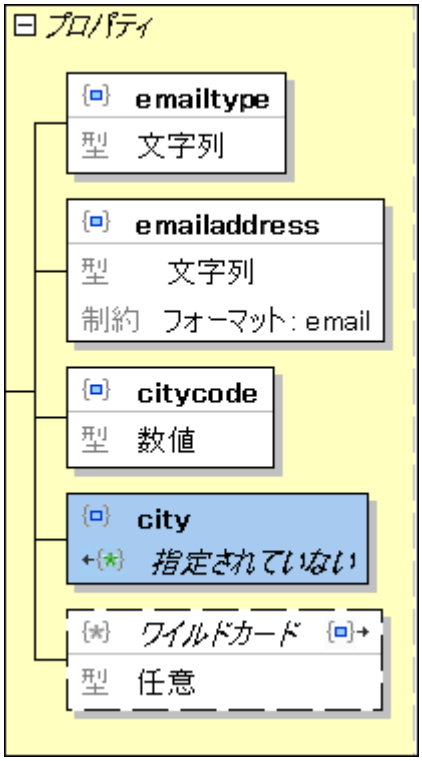
デザインビュー内では、スキーマで必要とされているにもかかわらず定義されていないため、指定されていないプロパティは赤でフラグ付けされています。JSON スキーマ自身は、有効ですが、検証に使用されるインスタンドキュメントが有効ではありません。これには以下の理由があります: (i) `city` プロパティが存在しない場合、`city` プロパティが必要なため、ドキュメントは無効になります。`city` プロパティが存在する場合、 `city` プロパティが定義されておらず、このプロパティのためのプロパティワイルドカードが許可されていないため、ドキュメントは無効になります (*下の暗黙的にプロパティを指定する* 517 *を参照*)。

指定されていないプロパティのための定義を作成するには、以下を行います:

1. デザインビュー内の指定されていないプロパティを選択します。
2. 詳細入力ヘルパー内の指定されたチェックボックスをチェックします (*上のスクリーンショット参照*)。または、指定されているフラグをコンテキストメニューを使用して削除します。
3. 必要に応じてプロパティの定義を変更します。

暗黙的にプロパティを指定する

プロパティは、適切なパターンプロパティまたはプロパティワイルドカードを追加することにより暗黙的に指定することができます。下のスクリーンショットは、追加されたプロパティワイルドカードを表示しています。 `city` と名前の付けられたインスタンスプロパティは、このワイルドカードに一致します。スキーマ内では、ですから、`city` プロパティは、ワイルドカードにより暗黙的に指定されたということができます。`city` プロパティを含むインスタンスファイルはこのスキーマに対して有効になります。



暗黙的に指定されているプロパティ内とプロパティワイルドカード内のそれぞれのアイコンに注意してください。各アイコンは他のプロパティへのリンクです。アイコンをダブルクリックするとの他のプロパティを選択します。

13.5.12.8 オブジェクトと依存性

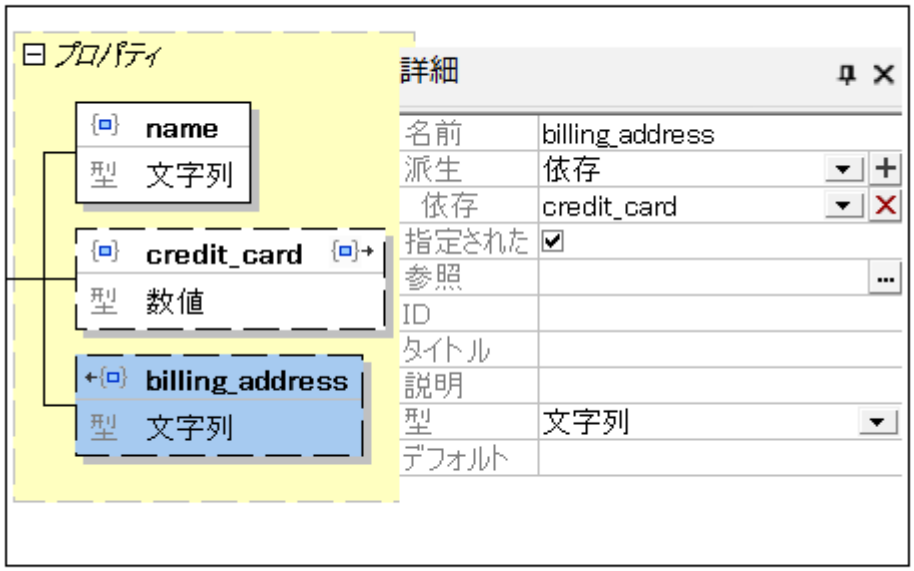
オブジェクトの定義では、特定のプロパティが他のプロパティが存在する場合のみ存在するように指定する必要がある場合があります。最初のプロパティは、2番目のプロパティの依存性となります。このセクションでは依存性を含むシナリオについて説明を行います。オブジェクト（例えば `member`）には `credit_card` という名前のプロパティがあるとします。オブジェクトの `billing_address` プロパティは、`credit_card` プロパティの依存性に設定することができます。これは、`credit_card` プロパティが存在する場合のみ、`billing_address` プロパティが存在することを意味します。

この種類の依存性は2通りの方法で指定することができます:

- プロパティ依存性として（依存性構造はプロパティです）
- スキーマ依存性として（依存性構造はスキーマです）

プロパティ依存性

下のスクリーンショットは、`name` プロパティ（必須）、`credit_card` プロパティ（任意）、および `billing_address` プロパティ（依存性）を持つオブジェクトを表示しています。`billing_address` プロパティは、依存性 `credit_card` プロパティにあります。JSON オブジェクト定義のコードは、下のスクリーンショットにリストされています。また、プロパティ依存性の作成方法は下に説明されています。



プロパティ依存性を持つ JSON オブジェクトのコードのリスト

```
{
   "type": "object",
   "properties": {
      "name": {
      "type": "string"
      },
      "credit_card": {
         "type": "number"
      },
      "billing_address": {
         "type": "string"
      }
   },
   "required": [ "name" ],
   "dependencies": {
      "credit_card": [ "billing_address" ]
   },
   "additionalProperties": false
}
```

プロパティ依存性を作成するには、以下を行います:

1. 依存性がベースにされるプロパティを右クリックします。（サンプルでは、`credit_card` プロパティです。）
2. 表示されるコンテキストメニューから [**依存性の追加 | 依存性プロパティ**] を選択します。新しいプロパティが依存性の値の発生に追加されます。
3. このプロパティの名前と値を定義し、追加の詳細または制約を追加します。

他のプロパティの依存性としてプロパティを指定するには、以下を行います:

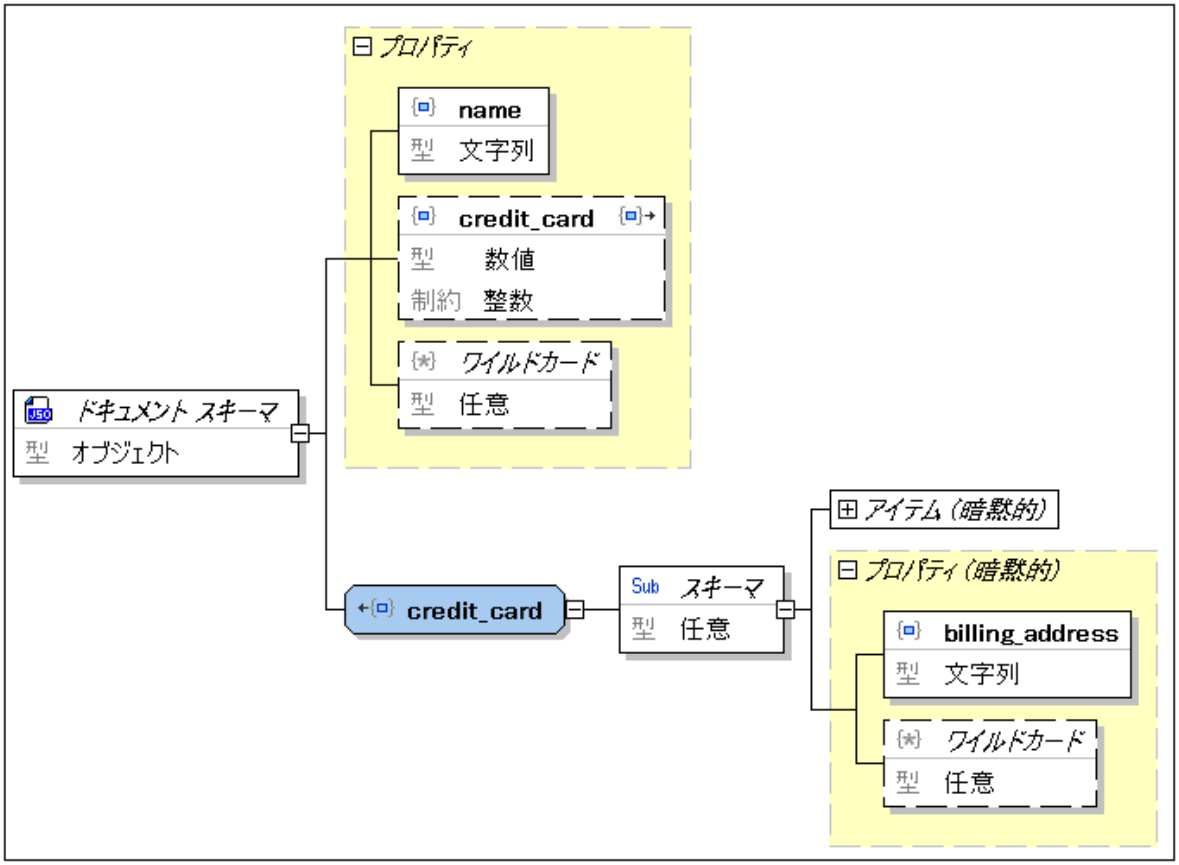
1. 他のプロパティの依存性とするプロパティを右クリックします。(サンプルでは、`billing_address` プロパティ)
2. 表示されるコンテキストメニューから、[**依存性**] を選択します。または、詳細入力ヘルパーで、出現の入力に移動して、*依存性* を選択します(*上のスクリーンショット参照*)。
3. 詳細入力ヘルパーで、入力の依存性のドロップダウンリストのアイコンをクリックします。ドロップダウンリストはオブジェクトのすべての他のプロパティを表示します。現在のプロパティが依存するプロパティを選択します。

メモ　依存性に含まれる両方のプロパティのボックスが表示されますアイコン(*上のスクリーンショット参照*)。プロパティのアイコンをダブルクリックすると他のプロパティに移動します。

メモ　プロパティは、複数の依存性プロパティを持つことができます。

スキーマ依存性

下のスクリーンショットは、前のセクションで同じインスタンスデータ構造をオブジェクトとして説明するオブジェクトについて説明しています。を表示します。2つのオブジェクトの定義は、しかしながら、異なります。前の定義は`billing_address` プロパティを`credit_card` プロパティ上の依存性と定義するためにプロパティ*依存性*を使用していますが、現在の定義は、*スキーマ依存性*をこの依存性を定義するために使用しています。後者のJSON オブジェクト定義のコードは下のスクリーンショットにリストされています。どのようにスキーマ依存性を作成するかは下に説明されています。



スキーマ依存性を持つJSON オブジェクトのコードリスト

スキーマ依存性を作成するには、以下を行います:

1. 依存性がベースにするプロパティを右クリックします（サンプル内では credit_card プロパティです）。
2. 表示されるコンテキストメニューから「**依存性の追加 | スキーマ依存性**」を選択します。新しいオブジェクト定義が作成されます。オブジェクトは、依存性のプロパティと同じ名前（サンプル内では credit_card）を持ち、子サブスキーマを持ちます。
3. サブスキーマを定義して、追加詳細または制約を追加します。

メモ 依存性に関連するプロパティおよびオブジェクトのボックス内にアイコンが表示されます（*上のスクリーンショット参照*）。ボックス内のアイコンをダブルクリックすると他のボックスに移動することができます。

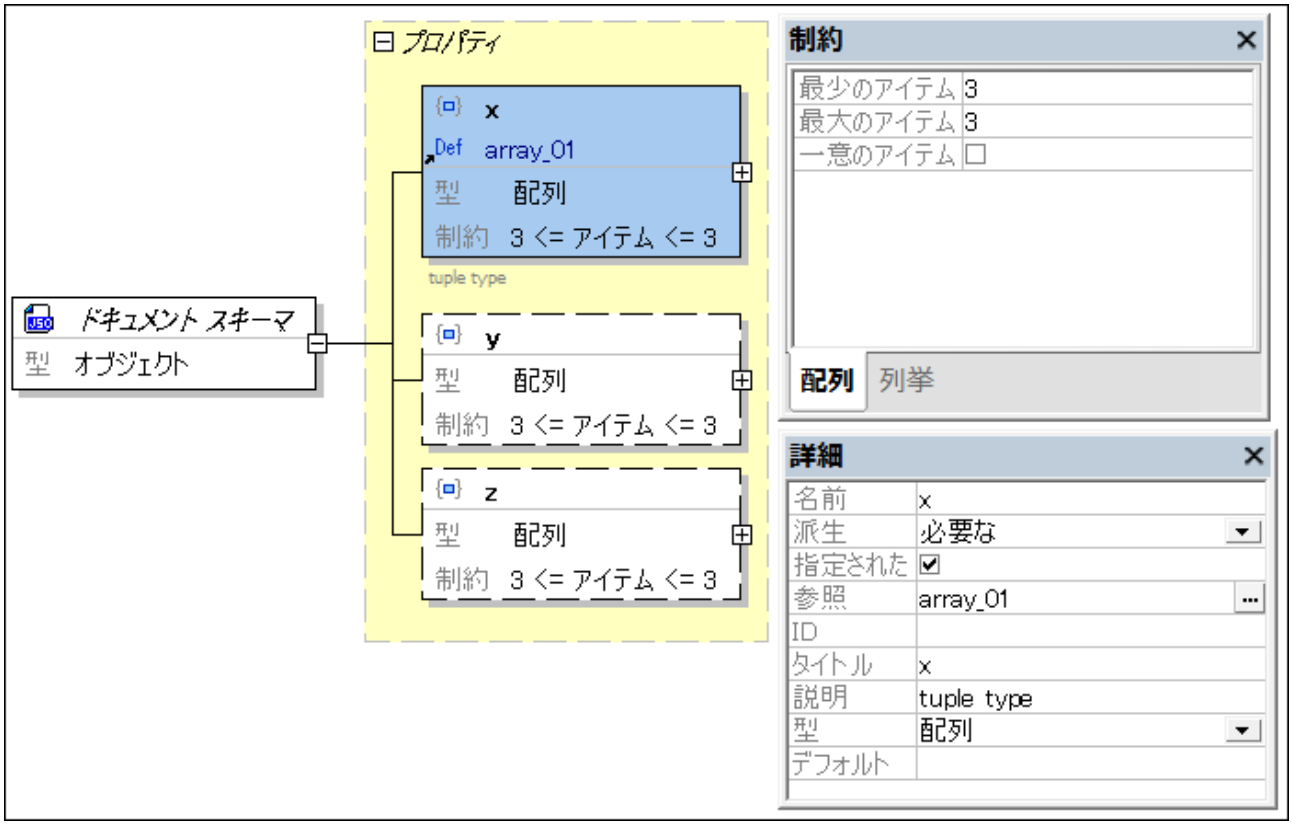
メモ 複数の依存性を設定するには、依存性 サブスキーマ 内で行います。（*上のスクリーンショット参照*）。

13.5.12.9 配列

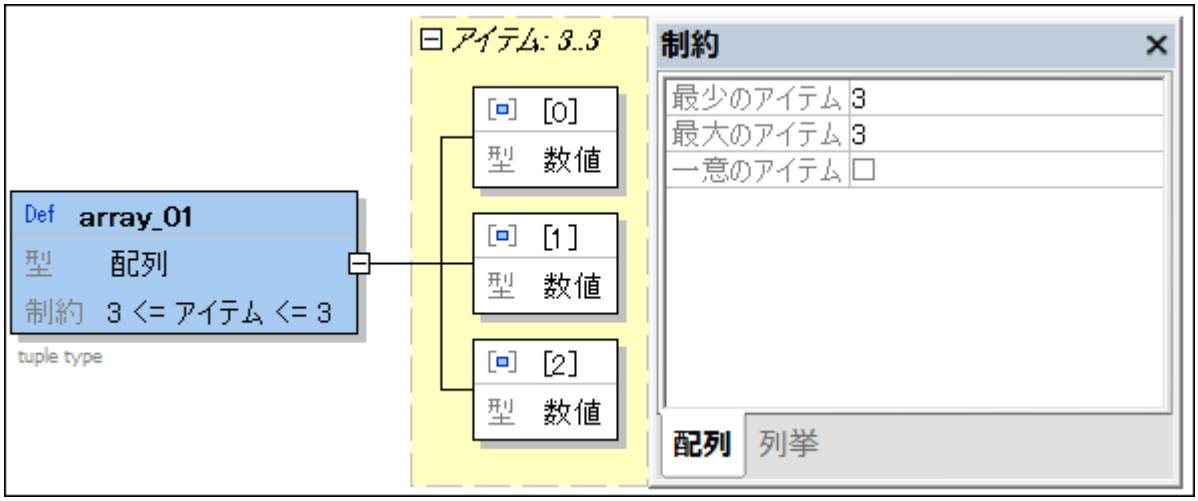
配列は、順序付けされたゼロまたは複数のアイテムで、角かっこにより区切られています。リスト内の各アイテムには、型が割り当てられています。下のインスタンスのリストは、3 つのプロパティを持つオブジェクトです。各プロパティの値は（黄色のハイライトで区切られている）配列です。

```
{
      "x": [ 1, 2, "abc" ],
      "y": [ 3, 4, "def" ],
      "z": [ 5, 6, "ghi" ]
}
```

上のリスト内の 3 つすべての配列は、同じ定義を持ちます。それぞれ、3 つの順序付けされたアイテムを次の順番で含んでいます: (i) 数値アイテム、(ii) 数値アイテム、(iii) 文字列アイテム。このオブジェクトのスキーマの詳細は、下のスクリーンショットに表示されています。3 つの配列すべての定義は同じなため、定義は、array_01 という名前のグローバル配列内で作成されています。3 つの配列（x、y、および z）のそれぞれは、以下のグローバル配列を参照しています: 512 array_01。



上のスクリーンショットでは、（青いハイライトで表示されている）配列 x が選択されています。詳細および制約は、それぞれ入力ヘルパーに表示されています（*上のスクリーンショット参照*）。許可されているアイテムの制約の数に注意してください。数は、制約入力ヘルパーで編集することができ、ダイアグラム内で表示されています。配列 アイテムは、配列自身の定義内で定義することができます。この場合は、グローバル定義 `array_01`（*下のスクリーンショット*）です。



以下の点に注意してください：

- `unique` 制約は、配列内のすべてのアイテムが一意であることを指定します。

- アイテムの番号付けは 0 から開始されます。
- ダイアグラム内のフレーズ、*3 <= アイテム <= 3* および *アイテム: 3..3* (*上のスクリーンショット参照*)、は、許可される最小および最大のアイテムを表しています。この場合、3 つのアイテムが存在する必要があります。

配列アイテム、配列アイテムワイルドカード、contains キーワードの追加方法

配列アイテム、配列アイテムワイルドカード、および配列の `contains` キーワードは定義、または、配列アイテムのコンテキストメニューを使用して追加することができます。配列ワイルドカードにより、配列内に含まれるオブジェクトの範囲を広くすることができます。`contains` キーワードは一方、`contains` キーワードの値が有効な JSON スキーマであること、および、配列の要素の少なくとも 1 つが与えられたスキーマに対して有効であることを指定します。

13.5.12.10 アトミック型

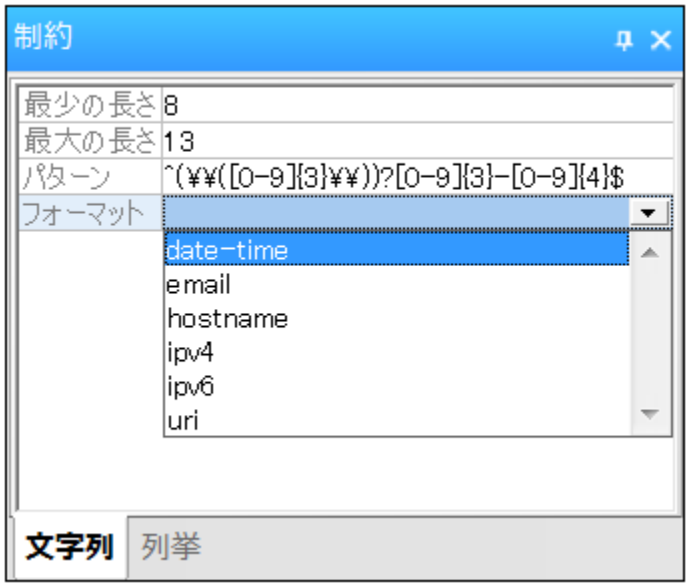
5 つの JSON アトミック (単純型 または プリミティブとして知られる) 型があります: (i) `string`、(ii) `number`、(iii) `integer`、(iv) `boolean`、および (v) `null`。これらのアトミック型のうちの 1 つの定義を指定するには、、以下のうち 1 つを行います:

- 定義のボックス内の *型* 値フィールドをダブルクリックして、型を選択します。
- 詳細入力ヘルパー内で、*型* フィールド内のドロップダウンリストから型を選択します。

各アトミック型の制約は、下に説明されています。

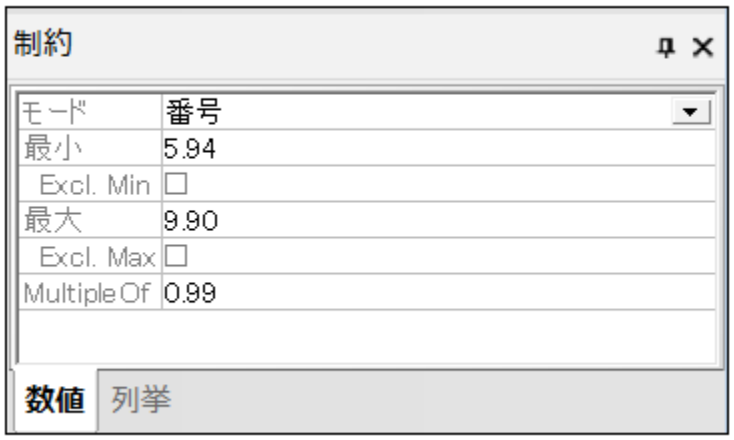
文字列

文字列 型に関しては、以下の制約を指定することができます: (i) 文字列の長さ、(ii) 文字列のパターンを説明する正規表現、(iii) 仕様からの定義済みのフォーマット。



数値

数値 型は、2つの型（数値 および 整数。 *下のスクリーンショット参照*）のための総合的な名前です。実際の型は、モードフィールド（ デフォルトは数値）で設定されます。2つの型の間の違いは、数値型は小数点を許可しますが、整数型は許可しない点です。*MultipleOf* フィールド内に値が存在すると、*インスタンス*値は、*MultipleOf* 値の整数である必要があります。



上の*スクリーンショット*で定義されている 数値 型の有効な値は、以下のとおりです: `5.94`、`6.93`、`7.92`、および `8.91`.

Boolean とNull

`boolean` 型は、値として `true` または `false` を取ります。 `null` 型 は、値として `null` を取ります。Neither 型 は任意の制約を取ります。

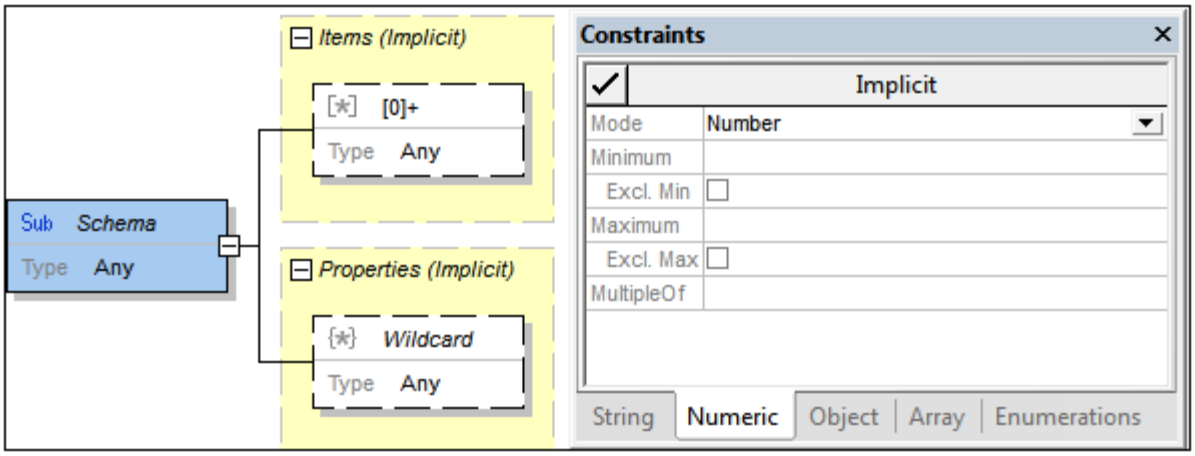
13.5.12.11 型セレクター（Any、Multiple）

JSON *スキーマビュー*の型コンボボックスのドロップダウンには、JSON 型: ではない、以下の2つの"型"があります: **`any`** および **`multiple`**。をリストします。これらは、型セレクターです。

- **`any`** 型セレクターは、任意のJSON 型を選択します。これは、*インスタンス*内で任意のJSON 型 がその特定の定義に対して有効であることを意味します。
- **`multiple`** 型セレクターは、1つまたは複数のJSON 型を選択します。これは、*インスタンス*が、スキマー内で選択されたJSON 型の1つである場合、*インスタンス*はその特定の定義に対して有効であることを意味します。
- (draft-06 内で新規の[506]) **`unconstrained`** 型セレクタ はJSON 型に制約を設定しません。これは、*インスタンス*内で、全てのJSON 型はその名前を持つ定義に対して有効です。
- (draft-06 内で新規の[506]) **`forbidden`** 型セレクタ はその名前を持つ定義の存在を効果的に許可せずJSON 型を禁止します。

any 型セレクター

any 型セレクターは、型が選択できるすべての箇所で選択されることができます。定義がスキーマに追加されると、 any がデフォルトの多々の選択になります。また、JSON 型 の1つが有効であることを指定します。これは、インスタンス型はオブジェクト、配列またはいずれのアトミック型（string、number、integer、boolean および null）である可能性があります。



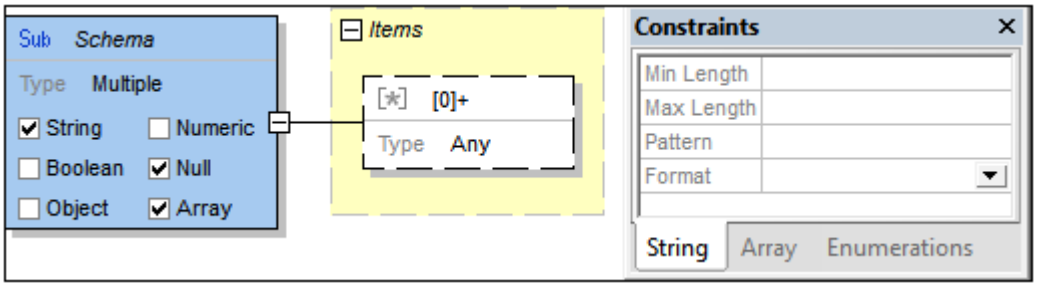
上のスクリーンショットでは、サブスキーマは Any の型を有します。ですから、すべての JSON 型 は、この定義に対して有効です。以下はユーザーインターフェイスに従い暗黙的に実装されています:

- オブジェクトが許可されているため、プロパティボックスは自動的に作成されます（*上のスクリーンショット参照*）。プロパティボックスは、デフォルトで、（Any の型のプロパティワイルドカードを使用して）任意の型の任意の数のプロパティを許可するように定義されています。プロパティの定義を自由に変更することができます。
- 配列が許可されているため、アイテムボックスは自動的に作成されます（*上のスクリーンショット参照*）。配列ボックスは、デフォルトで、（Any の型のアイテムワイルドカードを使用して）任意の型の任意の数のプロパティを許可するように定義されています。アイテムの定義を自由に変更することができます。
- 文字列と数値（数字と整数）は、許可されているため、これらのアトミック型のための制約は、制約入力ヘルパー内で定義することができます。

これらの型すべては、ですから、暗黙的に、 Any 型セレクターにより定義されています。特定の型に型を変更する場合はその型を選択してください。オブジェクトと配列を指定するには別の方法があります: オブジェクトまたは配列を右クリックして、**明示的に設定**を選択します。この方法は、選択された型を型とし、他の型を削除、またはオブジェクト/配列型を非アクティブにします。

multiple 型セレクター

multiple 型セレクターは、型が選択される箇所すべてで選択されることができます。許可する型をチェックすることにより、1つまたは複数の JSON 型の選択を許可します（*下のスクリーンショット参照*）。選択された制約入力ヘルパー内の型のための制約を指定することができます。



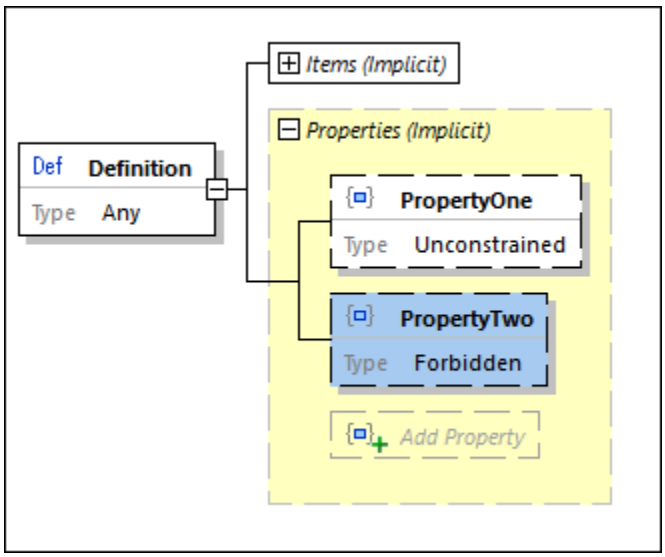
上のスクリーンショットでは、サブスキーマは、`string`、`null`、および`array` の型を許可します（*スクリーンショット参照*）。

- 文字列の制約は入力ヘルパーの制約で定義されています。
- null 型は、furr 制約をとりません。
- 配列アイテムボックスが自動的に作成されます。許可された配列アイテムの数値と型を定義することができます。

インスタンスドキュメントでは、選択された型はサブスキーマに対応する箇所に許可されています。

制約がなく禁止されている型セレクタ

`unconstrained` と`forbidden` 型セレクタは型を選択できる箇所で選択することができます。これにより、全ての型のオブジェクトが許可されている、または、その名前のオブジェクトが許可されていないかをそれぞれ指定することができます。

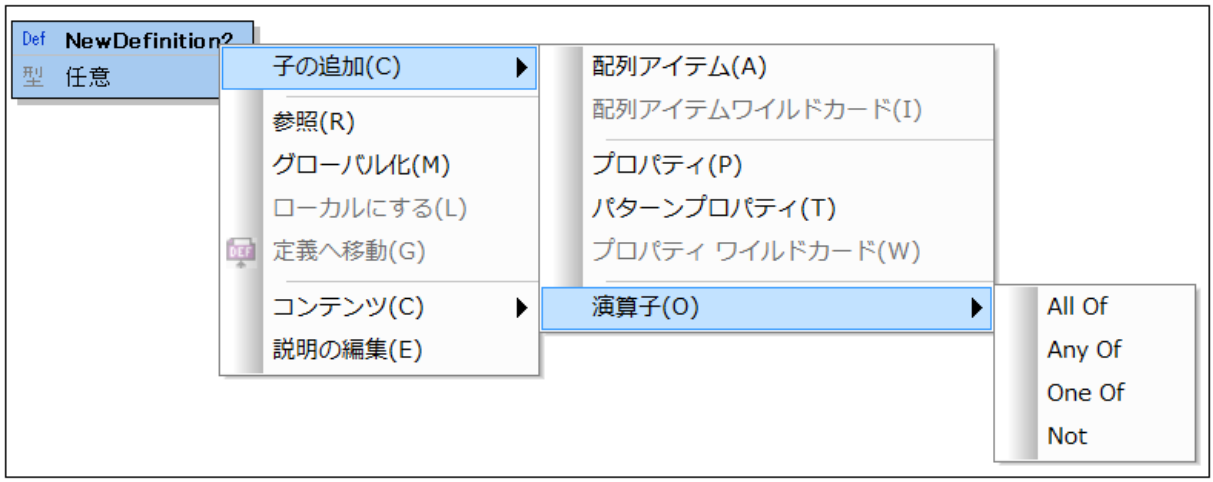


上のスクリーンショットでは、定義には2つのプロパティが存在します。**`PropertyOne`** はすべての型を持つことができますが、**`PropertyTwo`**という名前のプロパティは許可されていません（*スクリーンショットを参照してください*）。テキストの書式では、これは以下のようなコードリスティングを作成します。

```
"Definition": {
	"properties": {
		"PropertyOne": true,
		"PropertyTwo": false
	}
}
```

13.5.12.12 演算子

以下の4つの演算子が存在します: (i) `allOf`、(ii) `anyOf`、(iii) `oneOf`、および(iv) `not` 。演算子は、下に説明されるように有効性の条件を指定します。任意の定義に演算子を追加することができます。演算子 サブメニューにアクセスするには、演算子を追加する定義を右クリックして、「**子の追加 | 演算子**」を選択します(*下のスクリーンショット参照*)。

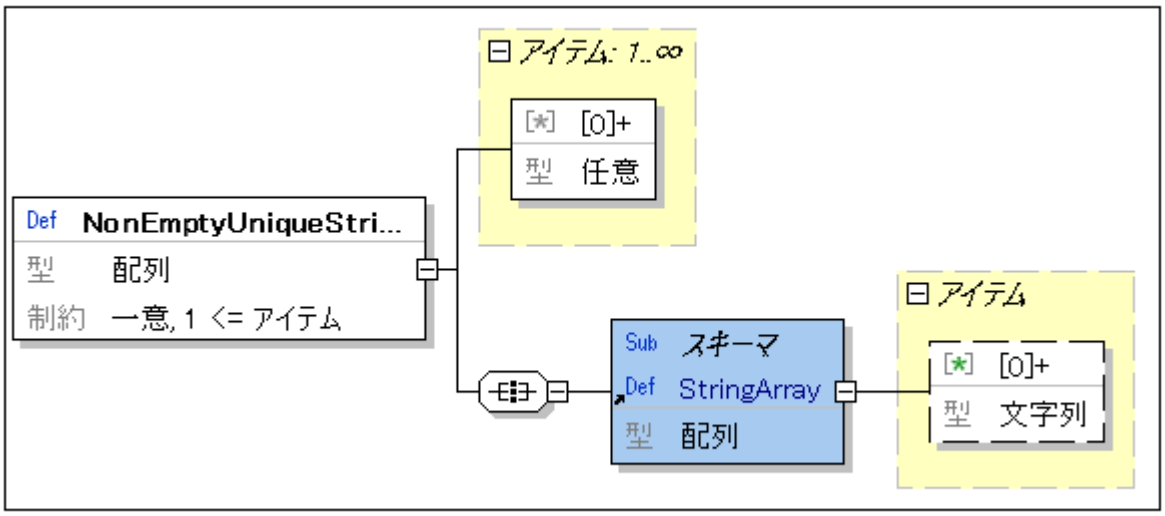


これらの演算子は、検証を成功するための条件を以下のように定義します:

演算子	アイコン	詳細
	All Of	値は、演算子の子として追加された1つまたは複数のサブスキーマ(定義)の配列です。*インスタンス*は、配列内のすべての定義に対して有効な場合有効です。
	Any Of	値は、演算子の子として追加された1つまたは複数のサブスキーマ(定義)の配列です。*インスタンス*は、配列内の1つまたは複数の定義に対して有効な場合有効です。
	One Of	値は、演算子の子として追加された1つまたは複数のサブスキーマ(定義)の配列です。*インスタンス*配列内の1つの定義に対して有効な場合有効です。
	Not Of	値は、演算子の子として追加されサブスキーマ(定義)たです。*インスタンス*は、与えられた定義に対して無効な場合有効です。

下のスクリーンショットは、3つのサブスキーマ(定義)の配列を子として持つ *All Of* 演算子を表示しています。*インスタンス*を有効にするには、(ドキュメント構造のこの箇所で) 3つのサブスキーマ定義の1つに一致する1つのJSON データ構造を持つ必要があります。

(C) 2015-2021 Altova GmbH

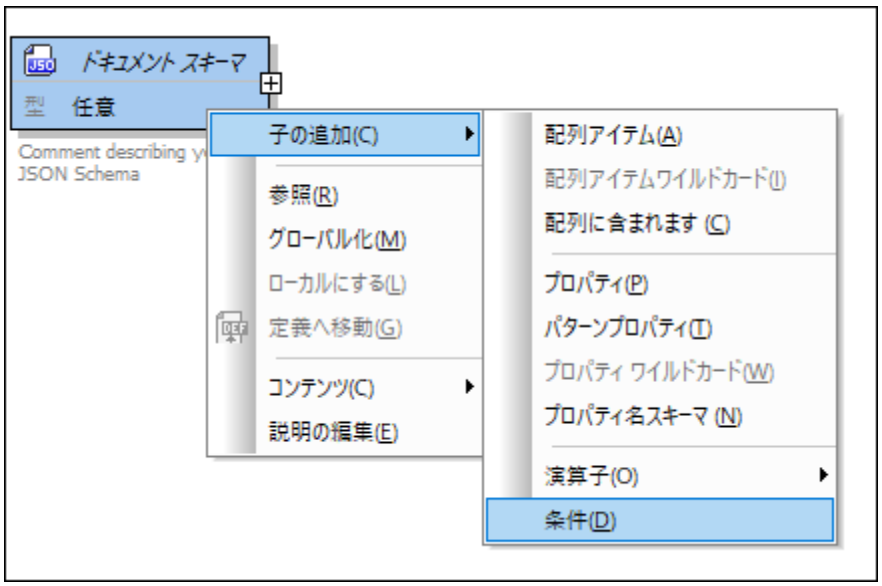


13.5.12.13 条件

条件は draft-07 506 の新規機能です。これにより、型および/または追加の型固有の制約などオブジェクトの特定のアスペクトにより制約を異なるものに指定できるようになります。

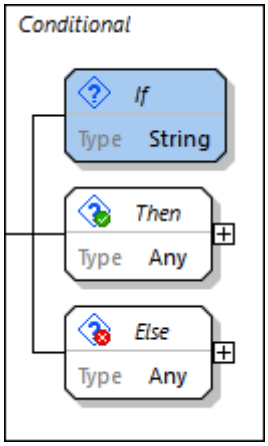
条件の追加

定義のコンテキストメニューを介して条件を定義に追加することができます（*下のスクリーンショットを参照してください*）。条件のサブメニューにアクセスするには、条件に追加する定義を定義を右クリックし、「**子の追加 | 条件**」を選択します。



条件の検証のセットアップ

条件は3つの要素を持つボックスとして追加されます: `If-Then-Else` (*下のスクリーンショットを参照してください*).

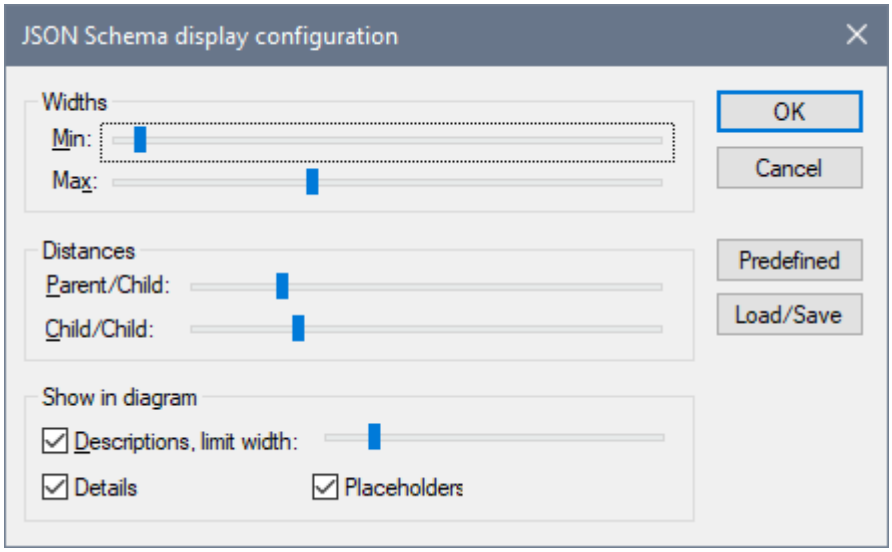


条件の検証をセットアップするには、以下を行います:

1. *If* ボックス内の条件を詳細入力ヘルパー内で最初に型を選択し、条件入力ヘルパー内の型ベースの条件を選択してセットアップします。
2. *Then* ボックス内で、(*If* ボックス内で指定されている) 条件が満たされていないイベント内の検証の必要条件をセットアップします。
3. *Else* ボックス内で、(*If* ボックス内で指定されている) 条件が満たされていないイベント内の検証の必要条件をセットアップします。

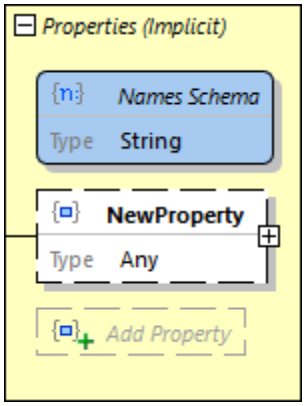
13.5.12.14 デザインビューの構成

メインウィンドウがデザインビューモードに設定されている場合、構成の表示ダイアログ (*下のスクリーンショット*) をメニューコマンド「**スキーマデザイン | ビューの構成**」を使用してアクセスすることができます。ここでデザインビューの外観を構成することができます。



デザインビューの次のアスペクトを構成することができます:

- *幅:* 2つのスライダーはそれぞれ、デザインビュー内のボックスの最小および最大の幅を決定します。また、ボックスの許可される幅の範囲を決定します。
- *親/子 の距離:* は、階層構造の各レベル間の左右の間隔を設定します。
- *子/子 の距離:* は、各ボックス間の上下の間隔を設定します。
- *詳細の幅:* は、詳細ラインの幅を設定します。テキストが幅をはみ出した場合、次のラインに折り返します。
- *詳細の表示:* 詳細ボックスのオプションをチェックまたはチェックを解除することにより、定義の詳細を表示または非表示に切り替えることができます。対応するツールバーアイコンが存在します。
- *プレースホルダの表示:* プレースホルダは定義されていないアイテムです。これらは可能性のあるアイテムを示しています。このオプションは、プレースホルダがオンに切り替えられているかを設定します。対応するツールバーアイコンが存在します。例えば、*プロパティの追加* アイテムは下のスクリーンショットではプレースホルダです。



メモ　「構成ビュー」メニューコマンドはオブジェクトの詳細な定義を表示するデザインビューモード 505 内でのみ有効化されています。定義概要グリッド 505 では使用することができません。

13.5.12.15 JSON スキーマドキュメンテーションの生成

IJSON スキーマアクティブなドキュメントの場合、「**スキーマデザイン | ドキュメントの生成**」コマンドをクリックすることでドキュメントを生成することができます。ドキュメントの出力は、HTML、MS Word、または RTF ファイルで生成することができ、含まれるコンポーネントを指定することができます。関連する JSON コンポーネントは、生成されたドキュメント内でハイパーリンクが付けられ、簡単に移動することができます。

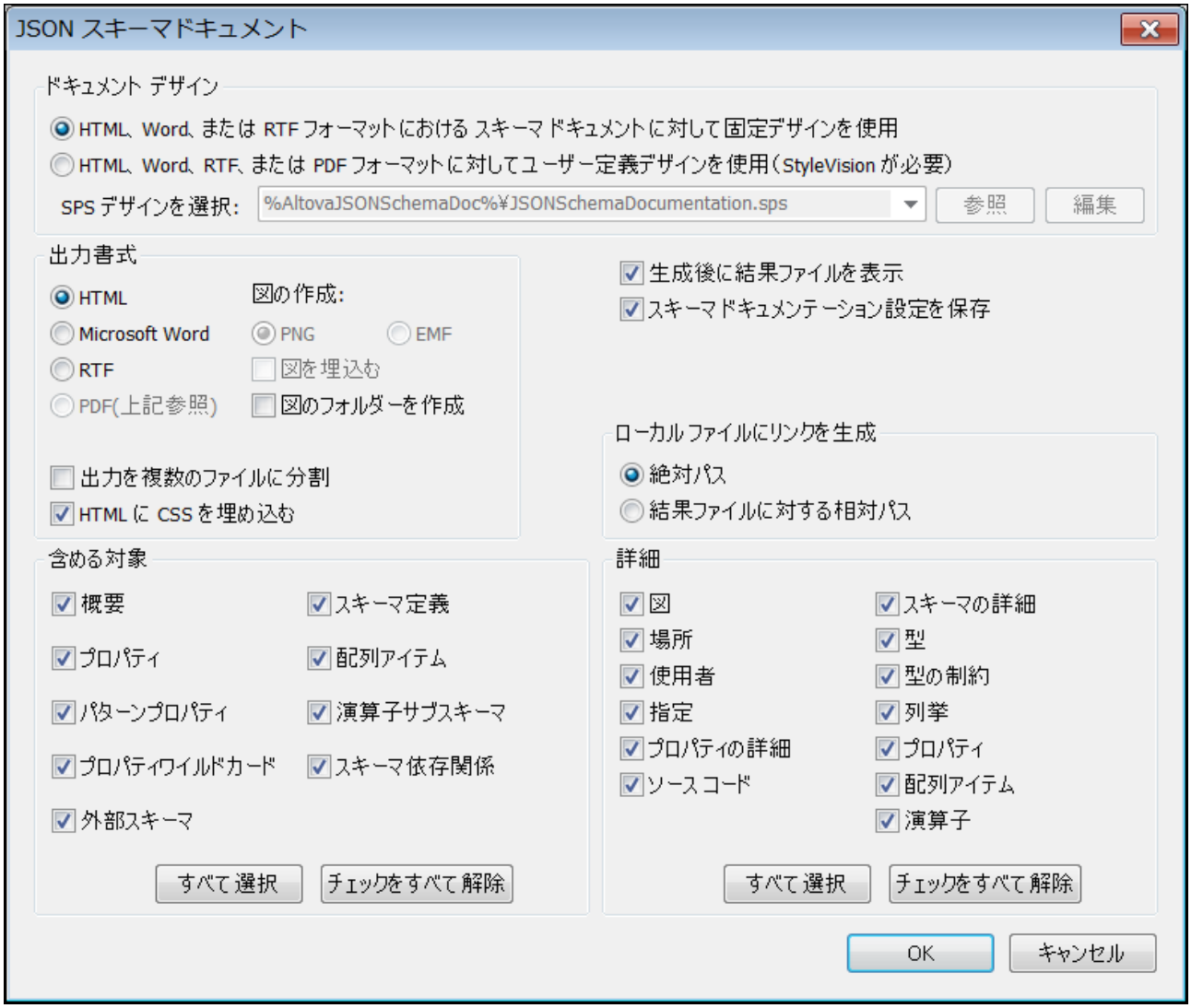
メモ MS Word フォーマットでドキュメントを生成するためには、MS Word（バージョン 2000 または以降）がインストールされている必要があります。

JSON スキーマドキュメンテーション生成のステップ

JSON スキーマファイルのためにドキュメントを生成するためには、以下を行います:

1. JSON スキーマがアクティブなドキュメントであることを確認します。
2. スキーマビューに切り替えます。
3. メニューコマンド「**スキーマデザイン | ドキュメントの生成**」を選択します。これにより、JSON スキーマドキュメンテーションダイアログボックスが開かれます（*下のスクリーンショット*）。
4. 生成する出力の型を次から選択します HTML、MS Word、または RTF。
5. ドキュメントに含まれる特別なコンポーネントと詳細を選択し、他のオプションを設定します（*下の JSON スキーマドキュメンテーションオプションを参照*）。

(C) 2015-2021 Altova GmbH



6. 「OK」をクリックして、JSON スキーマドキュメンテーションファイルの名前を、表示される名前をつけて保存ダイアログボックス内に入力します。

JSON スキーマドキュメンテーションオプション

次のドキュメントオプションから選択することができます:

- デザインテンプレート XMLSpy デザイン内でビルドイン(固定された)と指定することができ、またSPS ファイル内に保存されるユーザー定義デザインであることもできます。ユーザー定義 デザインの使用方法に関しては、ユーザー定義デザイン 872 のセクションを参照してください。
- 出力フォーマットペイン内で必要なフォーマットを指定することができます: HTML、Microsoft W ord、またはRTF。ドキュメントは単一のファイルまたは複数のファイルに生成されることができます。複数のファイルが生成される場合、それぞれのファイルは、コンポーネントに対応します。どのコンポーネントが出力に含まれるかは、含むペインのチェックボックスを使用して指定します。
- ダイアグラムの埋め込みオプションは、オMS W ord とRTF 出力 プションのために有効化することができます。このオプションがチェックされていると、ダイアグラムはPNG またはEMF フォーマットとして結果ファイル内に埋め込まれます。それ以外の場合は、ダイアグラムは、オブジェクトリンクを介した結果ファイル内にPNG またはEMF ファイルとして作成されます。出力がHTML の場合、全てのダイアグラムは、ドキュメント外部 PNG ファイルとして作成されます。

- 含むペイン内で、ドキュメントに含むアイテムを選択することができます。概要オプションは、ファイルの上にコンポーネント型別に整理された全てのコンポーネントをリストします。スキーマ定義が選択されていない場合、(外部スキーマ以外のすべての) 子コンポーネントは無効化されます
- 詳細ペインは、各コンポーネントに含まれる可能性のある詳細をリストします。スキーマ定義が選択されていない場合、全ての詳細は無効化されます。ドキュメントに含む詳細を選択します。
- 結果ファイルの表示オプションは全ての出力オプションのために有効化されています。このオプションがチェックされていると、結果はブラウザービュー(HTML 出力)、MS W ord (MS W ord 出力)、`.rtf` ファイルのデフォルトのアプリケーション(RTF 出力)に表示されます。

(C) 2015-2021 Altova GmbH

13.6 JSON ドキュメントの検証

XMLSpy は、以下を行うことのできる JSON 検証エンジンを搭載しています:

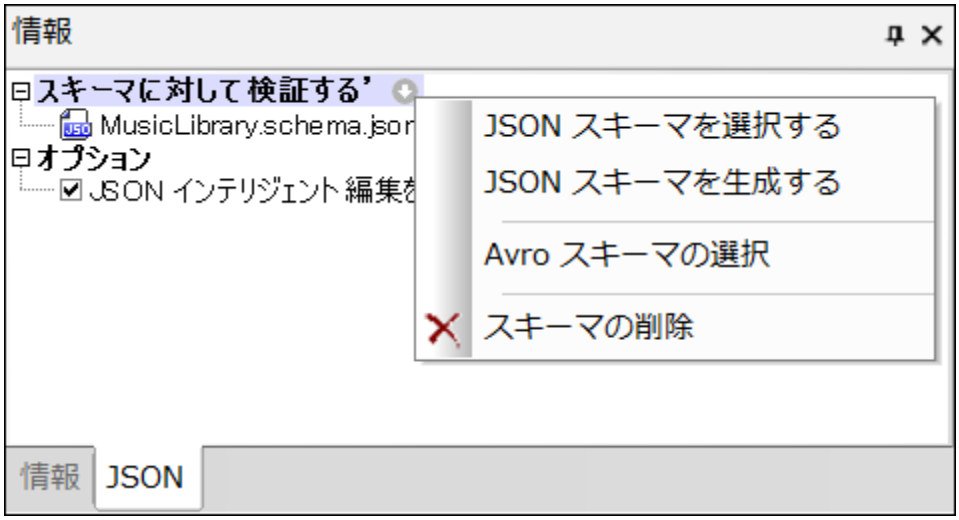
- *JSON スキーマがアクティブなドキュメントの場合:* JSON スキーマを適切な JSON スキーマ仕様に対して検証します(このため、追加のスキーマの割り当ては必要ありません)。スキーマバージョンは `$schema` 467 キーワードにより示されています。検証は、以下の3種類のビューで行うことができます(テキスト 470、グリッド 475、および JSON スキーマ 505)。
- *JSON インスタンスがアクティブなドキュメントの場合:* JSON インスタンスを JSON スキーマに対して検証します。スキーマは、JSON インスタンスに下で説明されるように割り当てられます。JSON インスタンス検証は、テキストビュー 470 とグリッドビュー 475 で実行されます。
- *JSON5 インスタンスがアクティブなドキュメントの場合:* JSON インスタンスを JSON スキーマに対して検証します。JSON5 インスタンスにスキーマが以下に説明されるとおり割り当てられます。JSON5 インスタンス検証はテキストビュー 470 とグリッドビュー 475 で実行することができます。

Avro 検証 (Enterprise Edition のみ)
Avro データと Avro スキーマドキュメントを JSON ドキュメントとして行われる検証は、テキストビュー 470 とグリッドビュー 475 で実行されます:

- *JSON フォーマットの Avro データインスタンスがアクティブなドキュメントの場合:* Avro インスタンスを Avro スキーマに対して検証します。スキーマは下で説明されるようにインスタンスに割り当てられます。
- *Avro スキーマがアクティブなドキュメントの場合:* Avro スキーマを Avro スキーマ仕様に対して検証します(スキーマの割り当ては必要ありません)。検証は、テキストビュー 470 または グリッドビュー 475 で実行されます。

JSON インスタンスに JSON スキーマを割り当てる

JSON インスタンスを JSON スキーマ、スキーマに対して検証するには、アクティブなインスタンスドキュメントにスキーマを割り当てる必要があります。アクティブな JSON インスタンスドキュメントの情報ウィンドウ(*下のスクリーンショット。Avro スキーマは、Enterprise Edition のみでサポートされています*)。または プロジェクトプロパティ 808 ダイアログを使用して、割り当てを入力します(*オプションつきの検証*)。JSON5 インスタンスドキュメントは、JSON スキーマに対してのみ検証されることに注意してください。



情報ウィンドウの JSON タブで、*スキーマに対する検証*の横のアイコンをクリックします、そして、表示されるメニューから、「JSON スキーマ」 をクリックします(*下のスクリーンショット。Avro スキーマは、Enterprise Edition のみでサポートされています*)。スキーマを参照して、「OK」をクリックします。スキーマがアクティブな JSON インスタンスドキュメントに割り当てられ、スキーマのファイル名が情報ウィンドウに入力されます。JSON インスタンスドキュメントが空の場合、JSON スキーマのインスタンスへの割り当ては、自動的に JSON インスタンスをスキーマをベースにしたサンプルデータで埋めます。

割り当てを解除するには、同じメニューから「**スキーマの削除**」を選択します（*上のスクリーンショット参照*）。

JSON *インスタンス*からの JSON *スキーマ*の生成に関する詳細は、JSON *インスタンス*からの JSON *スキーマ*の生成 545 のセクションを参照してください。

インスタンス および スキーマドキュメントの検証

アクティブな JSON ドキュメント（JSON *インスタンス/スキーマ*）を検証するには、コマンド「XML | XML の検証(F8)」を選択、または、ツールバー内の「**検証**」（F8）アイコン をクリックしてください。インスタンスドキュメントが検証される場合、スキーマドキュメントがインスタンスに割り当てれる必要があります（*上を参照*）。検証の結果はメッセージウィンドウに表示されます。エラーは、番号の付けられている余白内でもフラグが付けられます。エラーのためにスマートフィックスを使用することができる場合、電球アイコンがエラーを生成するライン内に表示されます。アイコンにマウスをポイントすると、使用することのできるスマートフィックスのリストのポップアップが表示されます。適用するフィックスを選択します。

メモ 上記の検証エラーインジケーターとスマートフィックスは **XML** | 「**検証**」（F8）コマンドが実行された場合のみ更新されます。これらは背景で更新されます。このため、エラーを修正した後、**検証**（F8）コマンドをもう一度実行し、エラーがフィックスされていることを確認する必要があります。

インスタンスドキュメントからスキーマドキュメンに移動するには、情報ウィンドウ内のスキーマをダブルクリックします（*上のスクリーンショットを参照*）または、コマンド DTD/スキーマ | **スキーマに移動** を選択します。JSON キーワードのスキーマ定義、または、オブジェクトに直接移動するには、インスタンスドキュメント内のキーワードまたはオブジェクトを選択し、DTD/スキーマ | **定義に移動**を選択します。

「**検証**」コマンドを使用して、JSON ファイルを含むプロジェクトフォルダー 676 を検証することもできます。

変更中に検証

編集中に検証 モードはデフォルトでオンに切り替えられています。オンに切り替えられていると、JSON グリッドビュー内でのドキュメントの変更中に整形式チェックと検証チェックが実行されます。（整形式のチェックに加え）JSON ドキュメントの検証を実行するには、JSON スキーマが JSON ドキュメントに割り当てられている必要があります。エラーはエラーの存在するテキストを赤色で表示し、エラーの箇所を赤い感嘆符で表示することで表示されています。詳細に関してはJSON ドキュメントの検証 537 を参照してください。

編集中に検証 モードは(i)「XML | 編集中に検証」 847 メニューコマンド（ii）**編集中に検証** ツールバーボタン または(iii)オプションダイアログの検証設定 1015 の *編集時* オプションによりオンとオフに切り替えることができます。

13.7 JSON フラグメントの挿入

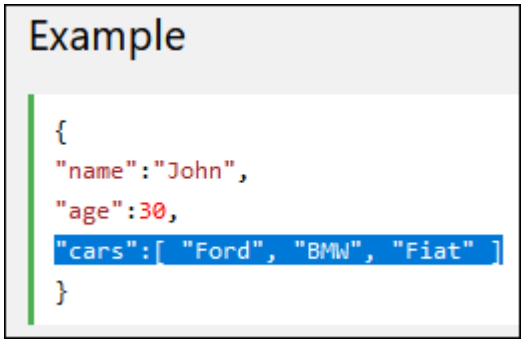
他のアプリケーションとWeb ページからJSON フラグメントを挿入することができます。2つの方法を使用してこれらのフラグメントを挿入することができます:

- ドラッグアンドドロップを使用してテキストビューとグリッドビューに挿入する。グリッドビューにドラッグアンドドロップする場合、ドラッグオーバーレイ内で使用可能なインテリジェントな情報 485 がフラグメントをドロップする箇所を決定する手助けをします。
- コピーして貼り付けを使用してテキストビューとグリッドビューに挿入する。

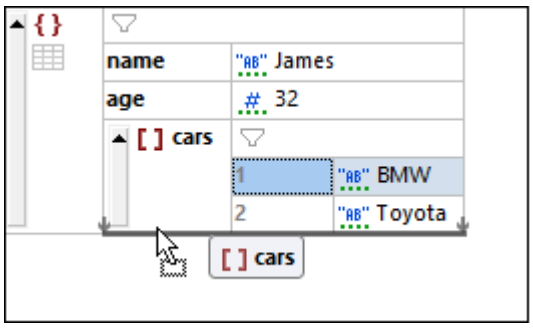
サンプル

次のサンプルはフラグメントをJSON ドキュメント内の正確な箇所に素早く追加する方法を示しています。

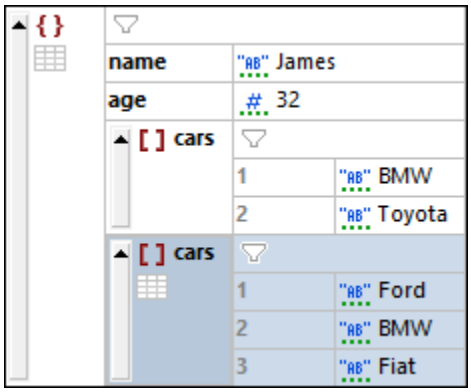
1. 下でハイライトされているフラグメントが(w3schools.com でのJSON チュートリアルから) 選択されています。`cars` という名前の配列



2. 下のスクリーンショットは類似している `cars` 配列を含むJSON ドキュメントのグリッドビューを表示しています。Web ページからのフラグメントが既存の `cars` 配列にドラッグされる場合 ドラッグされたJSON フラグメントが既存の配列の下に `cars` という名前の新規の配列としてドロップされる情報を含むドラッグオーバーレイ 485 が表示されます。



3. フラグメントがドロップされると希望する箇所に配置されます (*下のスクリーンショット*)。


{ }
name
"AB" James
age
32
[ ] cars
1
"AB" BMW
2
"AB" Toyota
[ ] cars
1
"AB" Ford
2
"AB" BMW
3
"AB" Fiat

13.8 XSLT/XQuery を使用した JSON 変換

XPath/XQuery 3.1 式を使用して JSON マップ、配列、および、オブジェクトをターゲットとすることができます。結果、JSON ドキュメントを、XSLT 3.0、XQuery 3.1、および XQuery Update 3.0 ドキュメントと共に XMLSpy の内蔵のエンジンを使用して変換することができます。

以下の機能を使用することができます:

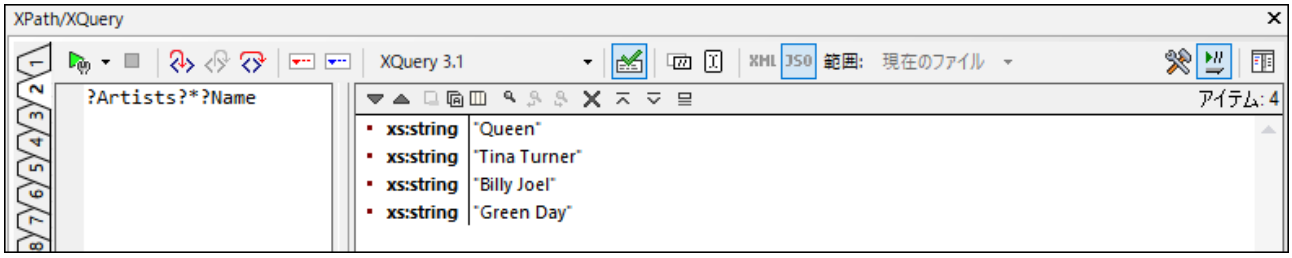
- XPath/XQuery 出力ウィンドウ[108] からアクティブな JSON ドキュメントを XPath/XQuery 3.1 式を使用してクエリ[541] することができます。
- アクティブな JSON ドキュメントをユーザーにより選択されている XSLT、または、XQuery ファイルを使用して変換[542] することができます。
- アクティブな XSLT、または、XQuery ドキュメントをユーザーにより選択されている JSON ソースファイル上で実行[542] することができます。

これらの機能は下記で詳細に説明されています。JSON のための XQuery 式作成ドキュメントに関する詳細は JSON のための XQuery 式[543] を参照してください。

メモ　アプリケーションフォルダー内の **Examples** プロジェクトの *JSON サンプル* フォルダー内の JSON、XSLT、および XQuery ファイルを使用して JSON を変換することができます: C:¥Documents and Settings¥<ユーザー名>¥マイ ドキュメント\Altova\XMLSpy2021\Examples.

XPath/XQuery 出力ウィンドウを使用して JSON ドキュメントをクエリする

JSON ドキュメントを XPath/XQuery 出力ウィンドウ[108] 内に XPath/XQuery 3.1 クエリ式を入力してクエリすることができます。(*下のスクリーンショットを参照*). **XPath 3.1** アイコン、または、**XQuery 3.1** アイコンを選択し、JSON 評価モード内にウィンドウが存在することを確認してください (*下記で説明されています*)。



下記の情報は JSON 評価モード内の JSON ドキュメントの評価に関しての情報です (XPath/XQuery ウィンドウの概要と使用の詳細に関しては次のセクションを参照してください: 出力ウィンドウ XPath/XQuery[108])。

JSON 評価モード
JSON 評価モードの以下の点に関して説明されています:

- XPath/XQuery ウィンドウは XML 評価モード、または、JSON 評価モードのどちらかです。現在アクティブな モードは、アクティブなモードのボタンがハイライトされることにより示されています。上のスクリーンショット内の XML/JSON 評価モード ボタンを参照してください。スクリーンショットでは、ウィンドウは JSON 評価モードに設定されています。
- 上のスクリーンショットでは、ボタンが無効化されていることを示す XML と JSON ボタンがグレーアウトされていることに注意してください。ボタンが無効化されると、有効化、または、無効化の状態を変更することはできません。反対にボタンが有効化 (グレーアウトされていません) されていると、ウィンドウの評価モードを変更することができます。
- XML/JSON 評価モード ボタンの*有効化/無効化*状態は評価のスコープにより異なります (*Where フィールドの値、上のスクリーンショット参照*)。評価スコープの値は、*有効化/無効化* 状態を決定するために 2 つのグループに分割されています: (i) 単一ファイル (*現在のファイル*)、(ii) 複数のファイル (*開かれているファイル、プロジェクト、フォルダー*)。

- (*上のスクリーンショットに表示されているとおり*) 評価スコープのために単一のファイル (*現在のファイル*) が選択されている場合、ウィンドウのモード (JSON、または、XML) はファイルの拡張子[1018] をベースにして決定されます。ファイルが JSON 適合[1018] の場合、JSON 評価モードは有効化されています。または、ファイルが JSON に適合しない場合[1018]、および、XML 評価モードがオンに切り替えられている場合。単一のファイルの型が既知なため、適切な評価モードが有効化されており、両方のボタンが無効化されているので、モードを変更することはできません。
- (*開かれているファイル、プロジェクト、フォルダー*) 複数のファイルオプションが選択されている場合、両方の評価モードボタンが有効化されており、ユーザーは有効化するモード (JSON、または、XML) を選択することができます。複数のファイルスコープのデフォルトの評価モードは XML です。
- XML 評価モードでは、XML 適合ファイル[1018] が処理され、JSON ファイルはスキップされます。
- JSON 評価モードでは、JSON 適合ファイル[1018] が処理され、XML ファイルはスキップされます。
- JSON 式をデバッグモード内でクエリすることもできます。

XSLT/XQuery を使用した JSON ドキュメントの変換

XSLT 3.0, XQuery 3.1、または、XQuery Update 3.0 ドキュメントを使用してアクティブな JSON ドキュメント[103] を変換するには、、以下を行います:

- *XSLT 3.0 変換:* 「**XSL/XQuery | XSL 変換**」をクリックし、を参照し、XSLT 3.0 ファイルを参照し、「**OK**」をクリックします。
- *XQuery 3.1、または、XQuery Update 3.0 変換:* 「**XSL/XQuery | XQuery/Update の実行**」をクリックし、XQuery 3.1、または、XQuery Update 3.0 ファイルを参照し、「**OK**」をクリックします。

変更済みのドキュメントが生成され、XMLSpy 内で直接確認することができます。

メモ XSLT/XQuery デバッガー[356] を JSON ドキュメントから開始することができますが、ブレークポイントとトレースポイントは XSLT、または、XQuery ドキュメント内でのみで設定します。

XSLT/XQuery ドキュメントのための JSON ソースの提供

アクティブな XSLT、または、XQuery ドキュメント[103] を JSON ソースファイル上で実行するには、以下を行います:

- *アクティブな XSLT 3.0 ドキュメント:* 「**XSL/XQuery | XSL 変換**」をクリックし、JSON ファイルを参照し、「**OK**」をクリックします。
- *アクティブな XQuery 3.1、または、XQuery Update 3.0 ドキュメント:* 「**XSL/XQuery | XQuery/Update の実行**」をクリックし、JSON ファイルを参照し、「**OK**」をクリックします。

変更済みのドキュメント変更済みのドキュメントが生成され、XMLSpy 内で直接確認することができます。

メモ XSLT/XQuery デバッガー[356] を XSLT、または、XQuery ドキュメントから開始することができ、JSON ドキュメントをデバッグセッションのための入力として割り当てることができます。ブレークポイントとトレースポイントは XSLT、または、XQuery ドキュメント内でのみで設定します。

13.9 JSON のための XQuery 式

JSON のための XQuery 3.1 式

JSON データ構造は通常オブジェクトと配列を使用するため、 XQuery 3.1 lookup operator `?` が（基本的に XQuery パースペクティブからマップする） JSON オブジェクトと JSON 配列内のノードを検索するために使用されます。ノードをロケートするこの方法は XML ドキュメント内でノードをロケートするために パス式が作成される方法とは異なります。これらの中では、スラッシュ演算子 `/` が パス式内でステップを接続するために使用されています（例: `items/*`）。JSON のための式 XQuery 内では、スラッシュ演算子はノードをロケートするために使用されません。

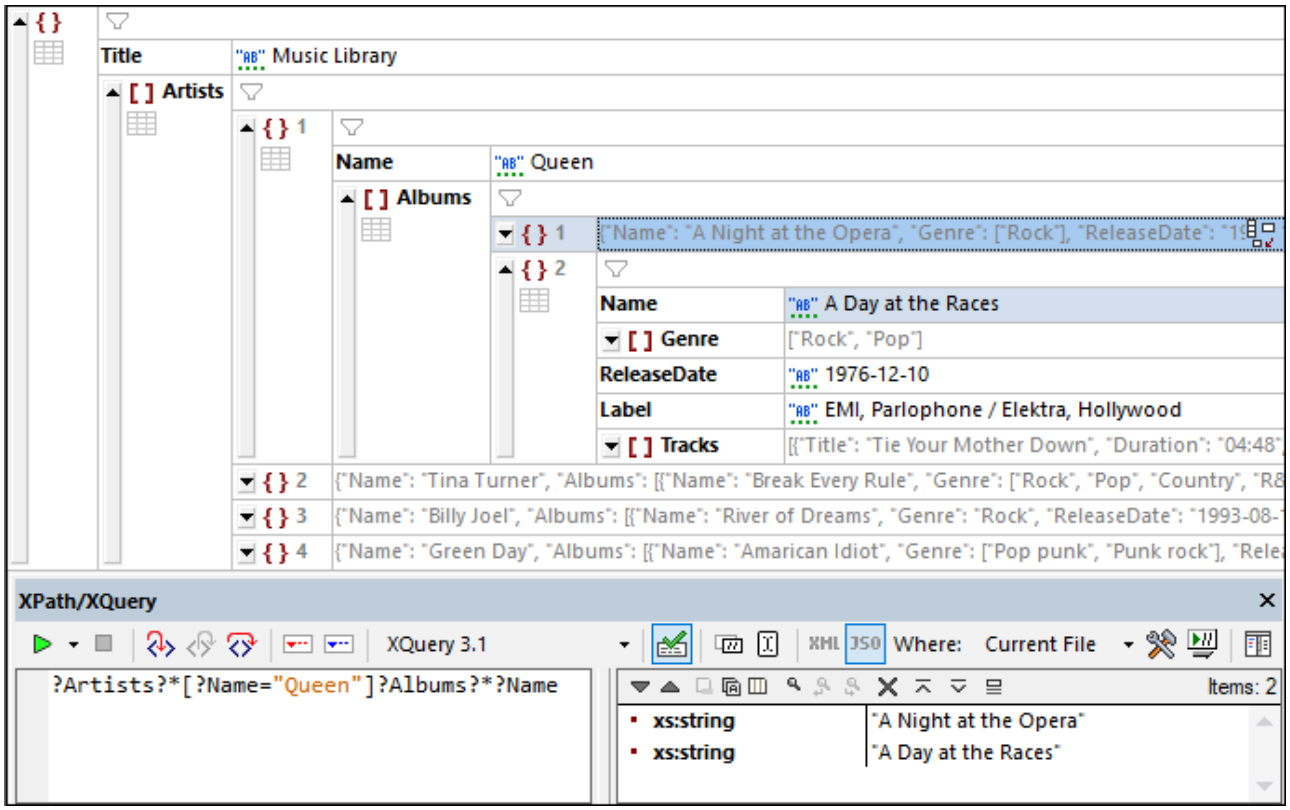
JSON のための XQuery 式のサンプル

`?items?*`
意味: 子ノード `items` をルックアップし、自信の子ノード全てをルックアップします。`items` はコンテキストノードの子ノードであることが期待されることに注意してください。

`?Artists?1?Albums?2?Name`
意味: 子ノード `Artists` をルックアップし、最初の子ノードをルックアップします。そのノードの内部で子ノード `Albums` をルックアップし、2 番目の子ノードをルックアップします。2 番目の `Name` ノードが返されます。

`?Tracks?*[contains(?Writer, 'Brian')]`
意味: 子ノード `Tracks` をルックアップし、自信のすべての子をルックアップします。子をルックアップ中に、各 `Writer` 子ノードの子をルックアップし、文字列 `'Brian'` を含むものを選択します。この式内には 3 つのルックアップ演算子が存在します。ノードセットがルックアップされる個所でそれぞれが新規ステップ内で使用されています。

`?Artists?*[?Name="Queen"]?Albums?*?Name`
意味: ルートオブジェクト内で lookup 子ノード `Artists` をルックアップし、`"Queen"` の値を持つ `Name` ノードを持つすべての子をルックアップします。コレアのノード内で、全ての子 Albums ノードをルックアップし、その子もルックアップします。これらの子内で、対応する `Name` ノードをルックアップ（して返）します。下のスクリーンショットでは、この式は XPath/XQuery ウィンドウ 381 内でターゲット JSON ドキュメントの JSON グリッドビュー 475 表示と共に表示されています。


Title
Music Library
Artists
Name
Queen
Albums
"Name": "A Night at the Opera", "Genre": ["Rock"], "ReleaseDate": "19
Name
A Day at the Races
Genre
["Rock", "Pop"]
ReleaseDate
1976-12-10
Label
EMI, Parlophone / Elektra, Hollywood
Tracks
[{"Title": "Tie Your Mother Down", "Duration": "04:48",
{"Name": "Tina Turner", "Albums": [{"Name": "Break Every Rule", "Genre": ["Rock", "Pop", "Country", "R&
{"Name": "Billy Joel", "Albums": [{"Name": "River of Dreams", "Genre": "Rock", "ReleaseDate": "1993-08-
{"Name": "Green Day", "Albums": [{"Name": "Amarican Idiot", "Genre": ["Pop punk", "Punk rock"], "Rele
XPath/XQuery
XQuery 3.1
Where: Current File
?Artists?*[?Name="Queen"]?Albums?*?Name
Items: 2
xs:string
"A Night at the Opera"
xs:string
"A Day at the Races"

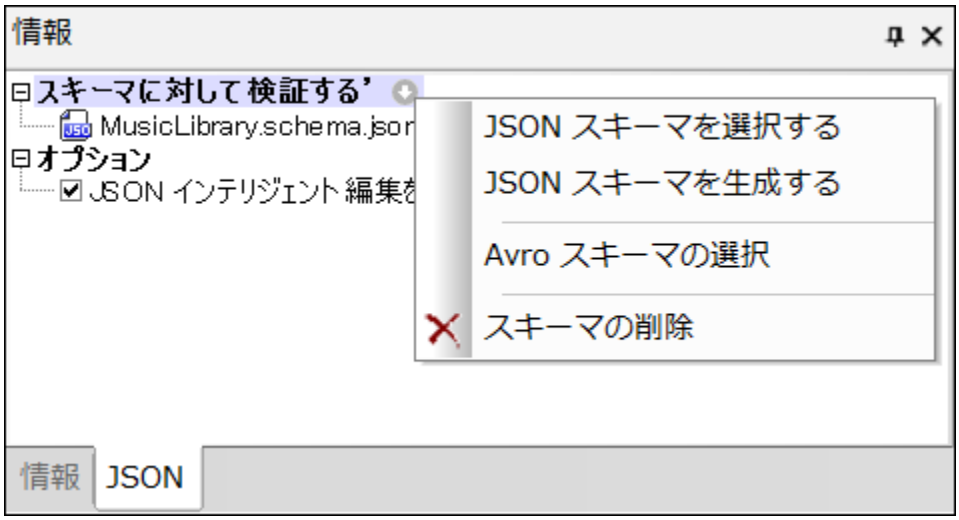
13.10　JSON インスタンスからの JSON スキーマの生成

XMLSpy は、(JSON5 *インスタンス*を含む) JSON *インスタンス*ドキュメントから JSON スキーマを生成することができます。この機能は、既存の JSON *インスタンス*をベースにしたスキーマを素早く提供し、スキーマをはじめから手動で作成する手間を省くため、とても役に立ちます。必要に応じて作成されたスキーマを修正することもできます。

スキーマの生成

JSON *インスタンス*から JSON スキーマを生成するには、以下を行います:

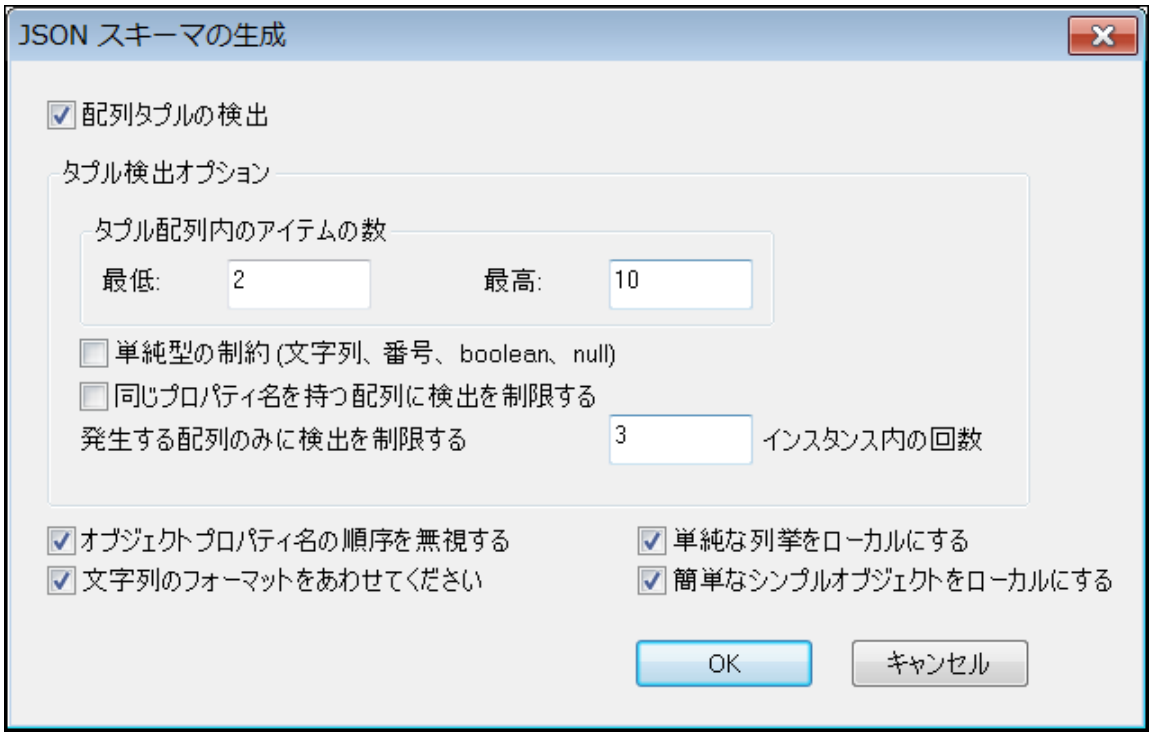
1. JSON *インスタンス*ドキュメントをアクティブなドキュメントにする。
2. 情報ウィンドウの JSON タブで、スキーマに対して検証するの横の矢印のアイコンをクリックすると、メニューが表示されます。「JSON スキーマの生成」をクリックまたは「DTD/スキーマ | DTD の生成/スキーマ」を選択します。



3. 表示された JSON スキーマの生成 ダイアログ(*下の次のセクションを参照*) 内で設定を変更します (*詳細は下を参照*)。編集が完了すると「OK」をクリックします。
4. 生成された JSON スキーマのための パスとファイル名を与えるようにプロンプトされます。「保存」をクリックすると、スキーマが生成されアクティブドキュメントになります。インスタンスドキュメント内で、生成されたスキーマファイルは検証で使用されるスキーマとして割り当てられます。以前に割り当てられたスキーマは上書きされます。割り当てを変更するには、コンテキストメニューの「JSON スキーマの選択」[537] コマンドを使用します (*上のスクリーンショット参照*)。

スキーマの生成のための設定

JSON スキーマの生成 ダイアログのスキーマの生成を指定することができます (*下のスクリーンショット*)。このダイアログにアクセスする方法の詳細に関しては、前のセクション[545]を参照してください。



配列タプルの検出
配列 タプルは配列内のアイテムのシーケンスです。例えば、次の配列には3 つのアイテムを持つタプルがあります: `[ 1, 2, "abc" ]`。配列の検証のためには、スキーマは配列(タプル)の順序とデータ型を考慮するか指定することができます。*配列タプルの検出* オプションがチェックされていると、(*上のスクリーンショット参照*)、アイテムの順序とデータ型が検出されます。検出された内容に従い、スキーマ内に対応する定義が作成されます。この設定のオプションは以下のとおりです:

- *タプル アイテムの数:* 指定することのできる最少および最大のタプルアイテムの数。インスタンス内のタプルがこの範囲内のアイテムの数を持つ場合、配列は検出され定義されます。
- *単純型のみ:* 単純型アイテム(アトミック型 `string`、`number`、`integer`、`boolean`、および `null`) を持つタプルのみが検出対象となります。
- *同一の名前を持つ配列:* 同じ名前を持つプロパティの値として定義された配列のみが検出対象となります。例えば、下の JSON データフラグメント内では、赤でハイライトされたかっこでマークされた配列は(青でハイライトされた) `a1` という名前のプロパティの値です: `{"object1": [ { "a1": [ 1, 2, "abc" ] }, { "a1": [ 3, 4, "def" ] }, { "a1": [ 5, 6, "ghi" ] } ] }`。
- *配列の最少の数値:* 指定することのできる配列の検出の最少の数です。

他の設定

- *オブジェクト プロパティ名前の順序を無視する:* 選択されると、プロパティのオブジェクトの順序がチェックされ、元に限りなく近く再作成されます。
- *文字列のフォーマットに一致させる:* 文字列データ型を指定するスキーマは特別なフォーマットを持つ必要があります。このオプションが選択されていると、XMLSpy は可能な箇所で文字列フォーマットを検出およびフォーマット定義を追加します。
- *単純配列をローカルに設定する:* 単純配列は、同じ単純データ型のすべてのアイテムの1 つです。選択されると、すべての単純配列は、ローカルで参照されているグローバル定義を使用する代わりにスキーマ内でローカルに定義されます。
- *単純オブジェクトをローカルに設定する:* 単純オブジェクトは、すべてのプロパティの値が同じ単純データ型をもつものです。選択された場合、ローカルで参照されているグローバル定義を使用するかわりに、すべての単純オブジェクトはスキーマ内で定義されます。

メモ　JSON スキーマが生成されると、個々のオブジェクトと配列の定義のローカル定義をグローバルにまたその反対に設定することができます。詳細に関しては、グローバルとローカル定義 512 のセクションを参照してください。

13.11 JSON インスタンスを JSON スキーマから生成する

テキストビュー、グリッドビューまたはスキーマビューで JSON スキーマがアクティブなファイルの場合、JSON スキーマから JSON インスタンスを生成することができます。「DTD/スキーマ | サンプル XML/JSON ファイルを生成する」860 をクリックします。このコマンドは、JSON5 ドキュメントではなく JSON ドキュメントを生成することに注意してください。

13.12 JSON と XML の間の変換

以下の変換オプションを使用することができます:

- XML インスタンスを JSON に変換 959: XML インスタンスドキュメントがアクティブなドキュメントの場合、JSON または JSON5 インスタンスドキュメントの生成を選択することができます。コマンド「変換 | XML インスタンスを JSON へ/から変換」959 を使用します。
- JSON インスタンスを XML に変換 959: JSON/JSON5 インスタンスドキュメントがアクティブなドキュメントの場合、XML インスタンスドキュメントは JSON インスタンスから「変換 | XML インスタンスを JSON へ/から変換」959 をクリックすることで生成されます。
- XML スキーマを JSON スキーマに変換 962: XML スキーマドキュメントがアクティブなドキュメントの場合、JSON スキーマドキュメントは XML スキーマから「変換 | XML スキーマを JSON スキーマへ/から変換」962 をクリックすることで生成されます。
- JSON スキーマを XML スキーマに変換 962: JSON スキーマドキュメントがアクティブなドキュメントの場合、XML スキーマドキュメントは JSON スキーマから「変換 | XML スキーマを JSON スキーマへ/から変換」962 をクリックすることで生成されます。

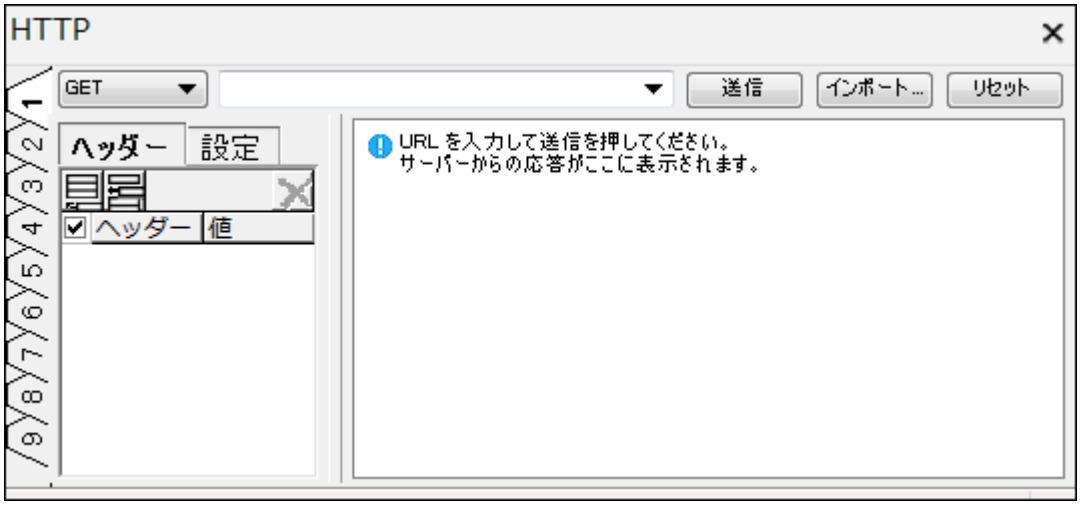
これらの変換はテキストビューとグリッドビューで有効化されます。上のリンクをクリックするとそれぞれの機能を確認することができます。

14 HTTP

HTTP（ハイパーテキスト・トランスファー プロトコル、略称 HTTP）はファイル（テキスト、イメージ、オーディオ、ビデオ、および他のマルチメディアファイル）のインターネット上での転送を定義する、プロトコール（または、ルールのセット）です。各 W eb サーバーはHTTP リクエストを継続的に待ち、受信後に処理をする（デーモンとして知られる）プログラムを実行します。例えば、W eb ホームページを訪問すると、ブラウザーはW eb サイトのW eb サーバーのホームページをリクエストするHTTP コマンドを送信します。サーバーのHTTP デーモンはリクエストを受信し、リクエストされたページを送信します。HTTP の重要なプロパティの1つは使用されません。すなわち、前の、または、後のコマンドを参照することなくHTTP コマンドが個別に実行されることを意味し、状態が無いことを表します。
XMLSpy 内では、HTTP 出力ウィンドウ内でHTTP コマンドをテストすることができます（*下のスクリーンショット*）。W eb サーバーにHTTP リクエストを作成、送信、受信し、更にレスポンスをチェックすることができます。

HTTP 出力ウィンドウのパーツ

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



ウィンドウは以下のパートにより構成されています:

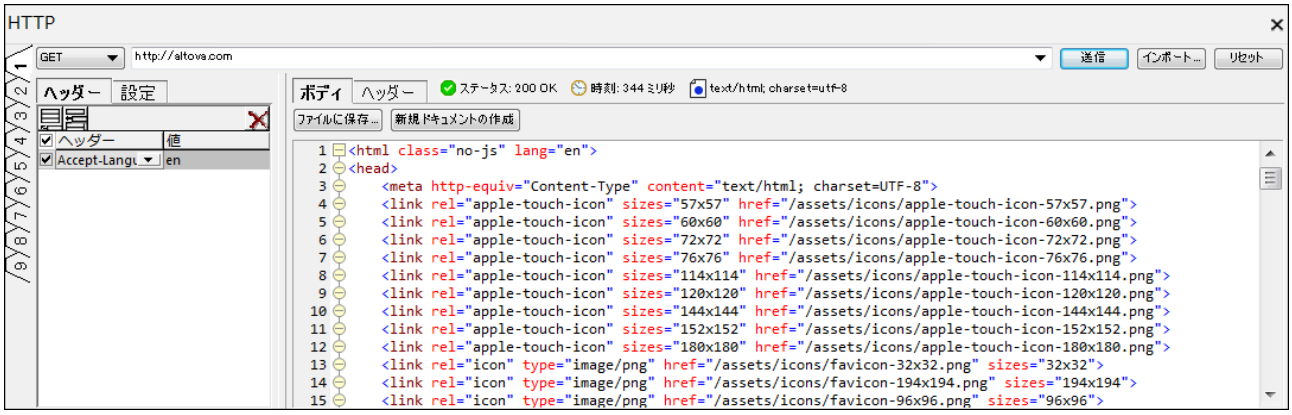
- 上の部分: (i) 使用するHTTP メソッドを選択するコンボボックス。(ii) W eb サーバーのURL のための入力フィールド。(iii) HTTP リクエストの実行に関連したボタン（「**送信**」、「**インポート**」、および「**リセット**」）。
- リクエストの作成 551 のための左側のペイン
- レスポンスの表示 557 のための右側のペイン

HTTP 出力ウィンドウの作動方法に関しては、このセクションのサブセクションで説明されています。

(C) 2015-2021 Altova GmbH

14.1　リクエストの送信

HTTP 出力ウィンドウ内でHTTP リクエストを送信することができます（*下のウィンドウ*）。**ウィンドウの左側のペイン** 内でリクエストは定義されます。ウィンドウの9つのタブそれぞれに異なるリクエストを設定することができ、それぞれは以下の部分から構成されています: (i)（ダイアログの上の部分で定義されている）リクエストとターゲット URL の HTTP メソッド[551]。(ii)（ヘッダータブ内の）リクエストのHTTP ヘッダー。(iii)（設定タブ内の）接続設定。(iv) 接続設定。および(iv)（下のスクリーンショットに表示されていないボディタブ）POST とPUT メソッド[551] の場合の HTTP メッセージボディ。（右上にある）「**リセット**」をクリックしてタブを空の状態に戻すことができます。



HTTP リクエストを送信するには、次をおこないます:

1. 左上のコンボボックスから HTTP メソッド（`GET`、`POST`、`PUT`、`DELETE`、`HEAD`、または `OPTIONS`）を選択します（*スクリーンショットを参照*）。
2. ターゲット W eb ページのURL を入力します（例、`http://www.altova.com`。`altova.com` のみを入力することもできます。URL の`http://` の部分は自動的に補完されます）。
3. ヘッダータブ内で、HTTP ヘッダー値を指定することができます（*上のスクリーンショットを参照*）。ヘッダーを選択、または、入力することができ、値を入力することができます（HTTP 1.1 ヘッダーのリストに関しては、ここを参照してください）。ツールバーのタブ内の「**挿入**」、「**追加**」、「**削除**」アイコンを使用して、ヘッダーを追加または削除してください。ヘッダーを削除する代わりに、ヘッダー名の左側の「有効化」チェックボックスをのチェックを解除して、ヘッダーを無効化することができます。これによりヘッダーを後で使用する場合、ヘッダーを再度入力する手間を省くことが出来ます。ヘッダーを受け入れる[558] も参照してください（送信時に自動的に追加されるヘッダーのために値を設定すると、自動的に追加する値の代わりにこの値を使用することができます）。
4. `POST` または `PUT` リクエストを送信する場合、ヘッダーと設定タブに加え、*ボディ*タブを使用できるようになります。ボディの作成方法に関しては、*POST とPUT リクエストのボディ*[552] のセクションで詳しく説明されています。
5. 設定タブ内でタイムアウトとセキュリティ設定を指定することができます。このタブに関する詳細は、下の*HTTP リクエストの設定*[553] のセクションを参照してください。
6. リクエストを送信するために、（ウィンドウの右上にある）「**送信**」をクリックします。
7. タブを空の状態に戻すには、「**リセット**」をクリックします。以下が発生します: (i) コンボボックスのドロップダウンリスト内の最初のメソッドを使用する（`GET`）。(ii) 現在のURL エントリが削除されます。(iii) 全てのヘッダー、設定、および、ボディの定義が削除されます。

メモ　ウィンドウの インポート[555] ボタンを使用してW ADL ファイルからHTTP 出力ウィンドウにリクエストをインポートすることができます。

メモ　UTF-8 エンコードでリクエストは送信されます。他のエンコードはUTF-8 に変換され、UTF-8 データが送信されます。

HTTP メソッド

次のHTTP メソッドがサポートされます

GET
GET メソッドは、指定されたURL にある*リソース*をリクエストします。URL にクエリを追加することもできます。例: `http://www.altova.com?name1=value1&name2=value2`。ヘッダーとボディを含むメッセージ内で*リソース*は返されます。

HEAD
HEAD メソッドとGET リクエストは類似していますが、メッセージボディは返されず、指定されたURL にある*リソース*について欲めた情報を含むメッセージヘッダーのみが返されます。

POST
指定されたURL にある既存の*リソース*を更新するために、または、指定されたURL で新規の*リソース*を作成するためにPOST メソッドが使用されます。*リソース*に提出するデータはHTTP リクエストのボディ内に置かれます。詳細に関しては、次を参照してください: *POST とPUT リクエストのボディ* 552。

PUT
PUT メソッド は、指定されたURL で新しい*リソース*を作成するために使用されます。*リソース*に提出するデータは、HTTP リクエストのボディ内に置かれます。詳細に関しては、次を参照してください: *POST とPUT リクエストのボディ* 552 。

DELETE
DELETE メソッドは、指定されたURL にある*リソース*を削除します。

OPTIONS
OPTIONS メソッド はサーバーがサポートするHTTP メソッドのリストを返します。

POST とPUT リクエストのボディ

POST とPUT リクエストのために、POST またはPUT リクエストのボディを指定することのできる追加 *ボディタブ*を使用できるようになります (*下のウィンドウを参照*)。*ボディタブには2 つのモードがあります*: **エディターモード** とファイルモード。ボディタブの左上のツールバーボタンをしようして、これら2 つのモード間を切り替えることが出来ます*スクリーンショットを参照*)。エディターモード内では、HTTP リクエストをペイン内で直接編集することができ、ファイルモード内ではHTTP リクエストのボディを含むファイルを選択することができます (*のスクリーンショット内で選択されている状態が示されています*)。

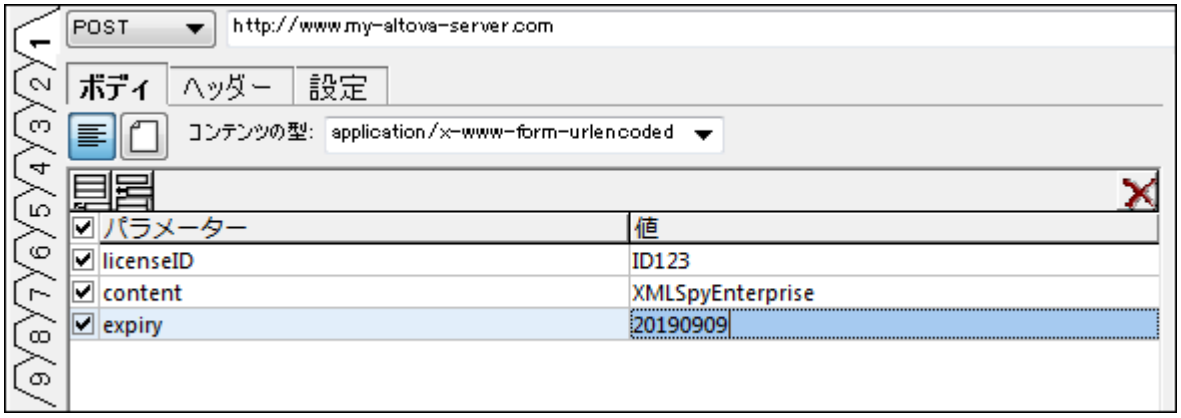
コンテンツ型フィールドによりリクエストのコンテンツの型 ヘッダーを指定することができます。このフィールドのコンボボックスオプションは各モード (エディターとファイル) により異なります。使用できるコンボボックスオプションから選択、または、MIME 型を入力します。このフィールドで指定されている値は、ヘッダー、または、ボディタブ内で指定されているコンテンツの型 ヘッダーを上書きすることに注意してください。

エディターモード
エディターのスタイルは選択されたコンテンツの型により異なります:

- **text/plain**、**text/xml**、**application/xml**、と**application/json** コンテンツ型: XML とJSON ドキュメントのための構文色分けや、行番号などのインテリジェントな編集機能を搭載したテキストエディター。上のスクリーンショットは

text/xml コンテンツ型のためのエディターを表示しています。リクエストのためのボディはエディター内に入力します。コンテンツ型フィールドにリクエストのコンテンツの型が指定されており、リクエスト以外の場所によるエントリにより上書きされることはありません。

- **application/x-www-form-urlencoded** コンテンツ型: 各ラインがリクエストのボディの名前値ペアを表示するグリッドビューを持つエディターです(*下のスクリーンショット*)。



ファイルモードを切り替えるには、ボディタブの左上にある「ファイル」アイコンを選択します(*上のスクリーンショットを参照*)。自動的にボディのコンテンツ型を有効化するには、コンテンツ型フィールド内で「自動」を選択します。コンテンツ型の自動的な検知はファイルの拡張子により決定されます。コンテンツ型を入力すると、入力されたコンテンツ型を使用してリクエストが送信されます。この場合、コンテンツ型が正確であることを確認してください。

リクエストのヘッダーと設定を(ヘッダータブと設定タブなどの)他のリクエストと同様簡素化することができます。

HTTP リクエストのための設定

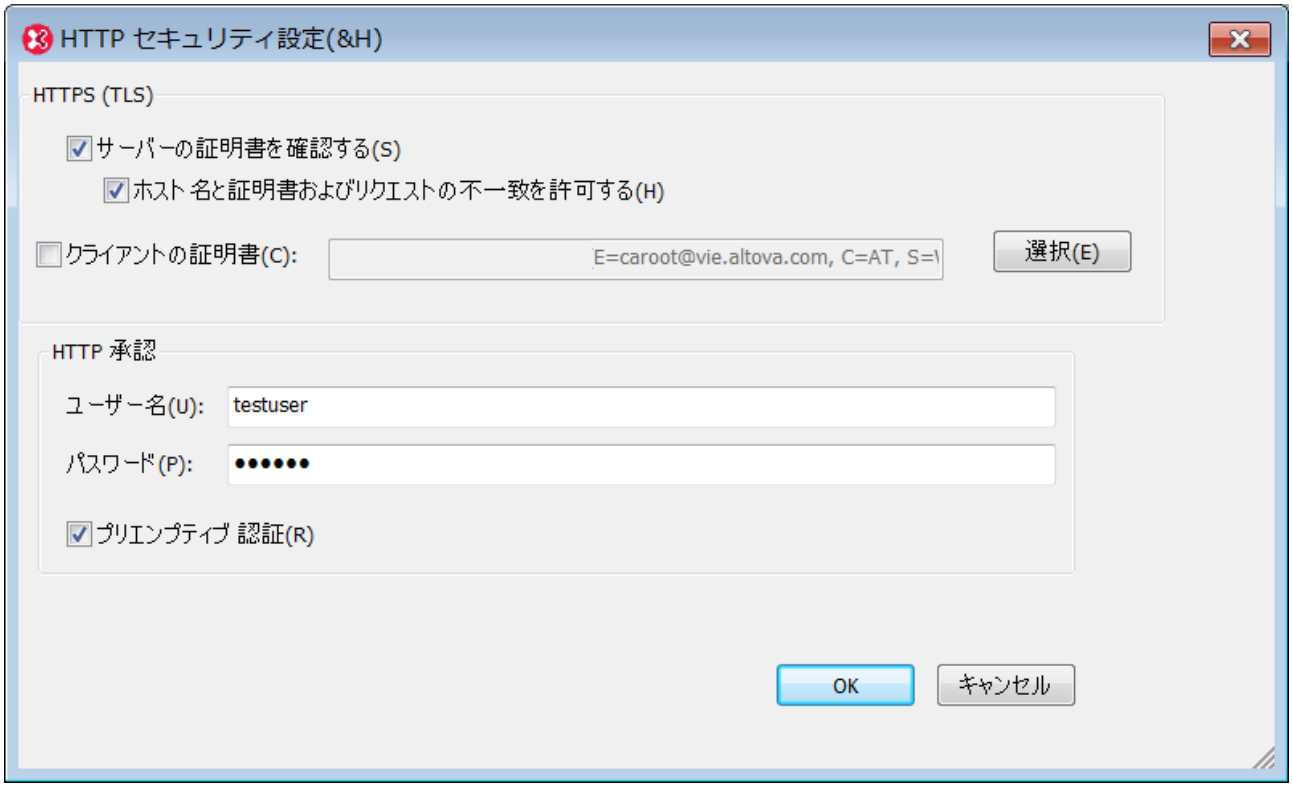
HTTP 出力ウィンドウの設定タブ内で次を設定することができます(i) 接続のタイムアウト、(ii) リクエストのセキュリティ設定(*下のウィンドウ*)。9つのウィンドウのタブのそれぞれに接続の設定を個別に設定できることに注意してください。下のスクリーンショット派 HTTPS URL のための設定を表示しています。

タイムアウト

XMLSpy がWeb サーバーと接続を試みる時間を秒数で指定することができます。接続が確立されずタイムアウトの時間に達すると、I/O Error 28: タイムアウト のメッセージが表示されます。タイムアウトの期間を指定しない場合、「無限」チェックボックスをチェックしてください。

セキュリティ設定

「編集」をクリックしてリクエストのセキュリティ設定を編集します。HTTP セキュリティ設定ダイアログが表示されます(*下のウィンドウ*)。HTTPS セキュリティ設定を指定し、そのタブにより作成されたリクエストのためのHTTP 認証資格情報を設定することができます。リクエストのターゲットのWeb サーバーがSSL を使用しない場合、HTTP 認証資格情報のみが使用されます。ターゲット Web サーバーが SSL を使用する場合、HTTPS セキュリティ設定とHTTP 認証資格情報が使用されます。



- *HTTPS セキュリティ設定:* デフォルトでは、サーバーの資格情報をチェックオプションがチェックされ、リクエスト内のホスト名が資格情報内のホスト名と異なる場合を指定することができます。クライアントの資格情報の使用を選択することもできます。(企業のネットワークなどの)、インターネット URL をターゲットする場合、(ローカル証明書ストア内にある) クライアント証明書をインターネットサーバー上の証明書を検証するために使用することがあります。
- *HTTP 認証:* サーバーへのリクエストの一部はユーザー認証を必要とする場合があります。このような場合、ユーザー名とパスワードをここに入力することができます。サーバーにより認証が必要とされる場合、認証は自動的に提供されます。それ以外の場合、サーバーが接続された後にプロンプトされる可能性があります。サーバーへの当初のリクエストが認証情報を含む場合、このプロセスはプリエンプティブ認証として称されます。プリエンプティブ認証がサーバーにより必要とされると、プリエンプティブ認証オプションを選択します。

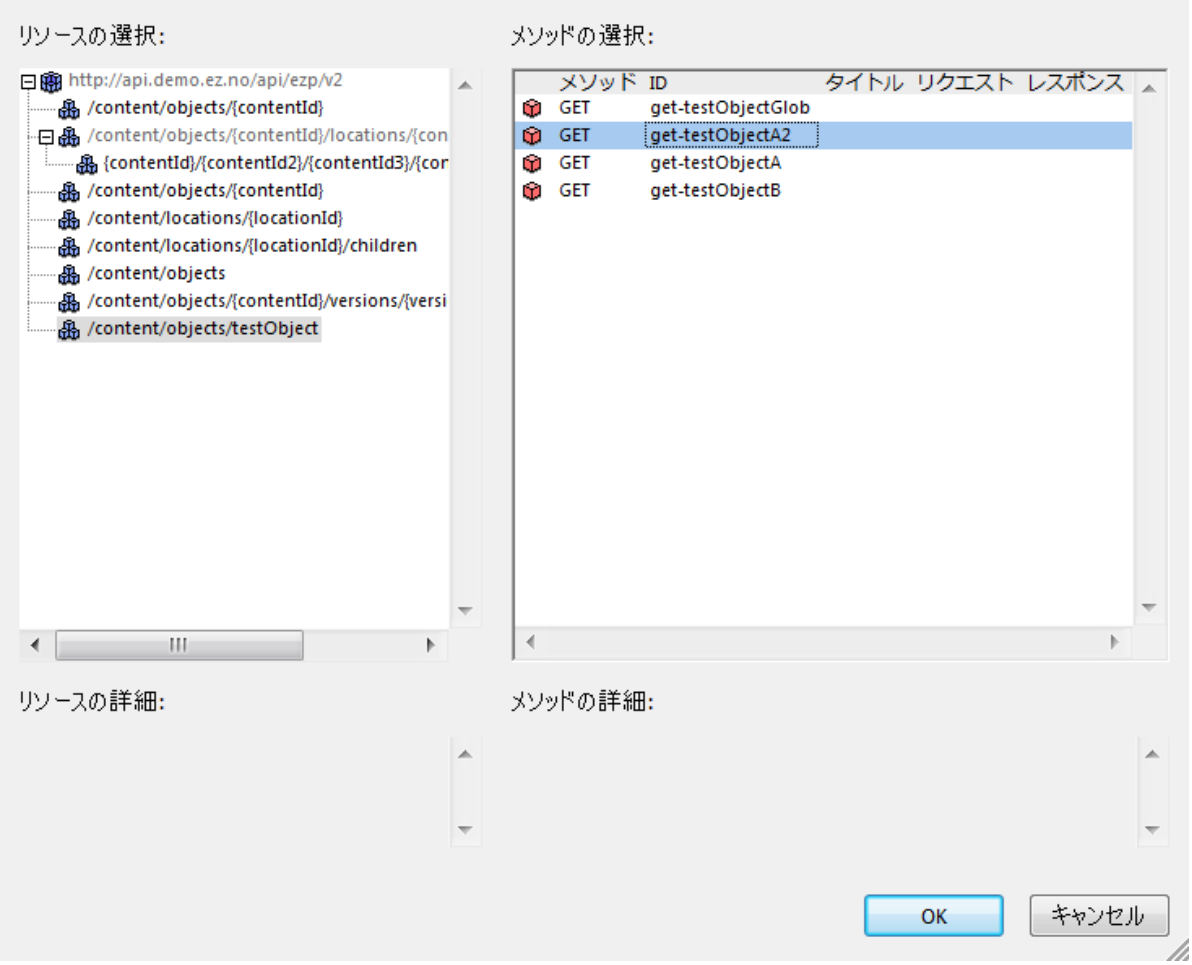
(C) 2015-2021 Altova GmbH

14.2 送信するリクエストのインポート

HTTP 出力ウィンドウ[550]内で、Web Application Development Language (WADL) ファイルからリクエストをインポートし送信することができます。これはXMLSpy のW ADL インポートウィザードを使用しておこなうことができます。ウィザードはW ADL ファイルを開きファイルのリソースからリクエストを選択することにより、リクエストの編集することの出来るパラメーターを編集できるようにし、HTTP 出力ウィンドウ[550]にインポートします。以下をおこないます:

HTTP 出力ウィンドウ[550] 内では、W SDL 1.1、W SDL 2.0、またはWeb Application Development Language (WADL) ファイルからリクエストをインポートし、送信することができます。これはXMLSpy のW SDL/W ADL インポートウィザードを使用して行います。ウィザードはW SDL またはW ADL ファイルを開き、ファイルのW SDL エンドポイント、または、W ADL リソースからリクエストを選択し、リクエストの編集可能なパラメーターを編集できるようにし、HTTP 出力ウィンドウ[550]にリクエストをインポートします。以下を行ってください:

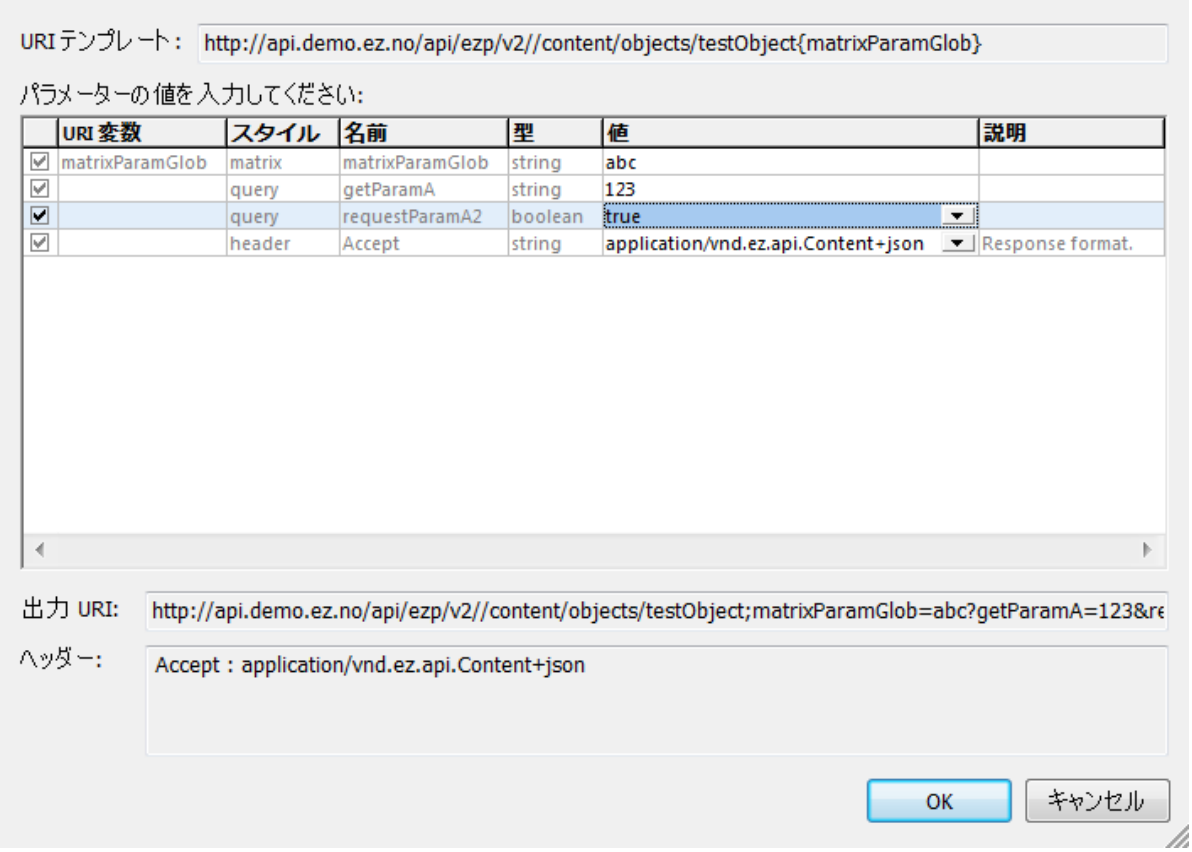
1. HTTP 出力ウィンドウ[550] 内で、W SDL/W ADL インポートウィザードを開始するためにインポート」ボタンをクリックします。
2. 表示されるファイル選択ダイアログ内で、インポートするリクエストを含むW ADL またはW SDL ファイルを参照し「OK」をクリックします。これによりW SDL/W ADL ウィザードのインポートが開始されます(*下のスクリーンショット*)。



3. 左側のペイン内で、(インポートするリクエストを含む)関連するW SDL エンドポイント、または、W ADL リソースを選択します。(*上のスクリーンショット参照*)。
4. 右側のペインでインポートする(W SDL オペレーション、または、W ADL メソッドである)リクエストを選択します(*上のスクリーンショット参照*)。W SDL を使用してインポートする場合以下の点に注意してください: (i) HTTP インポートはSOAP と

HTTP 拡張 のために提供されており、（ii）サポートされるバインディング（SOAP とHTTP）がこの（右側の）ペイン内に表示されます。

5. 「OK」をクリックします。リクエストにつまたは複数のパラメーターが含まれている場合、ウィザードの次のスクリーンは選択したリクエストのパラメーターを表示します（*下のスクリーンショット*）。それ以外の場合、リクエストはHTTP 出力ウィンドウ550にインポートされ、ウィザードは閉じられます。*下のポイント 7 を参照してください*。パラメーターはリクエストの一部です。検索リクエスト内では、例えば、1つのパラメーターが検索用語の場合があります。ウィザードはパラメーターの値をデータ型に対して検証し、3つの状態のいつを示します。ピンクの背景は、無効な値を示し、ベージュの背景はリクエスト内に入力される無効な値を示し、白い背景は有効な値を示しています。



6. HTTP リクエストパラメータースクリーン内で、必要に応じてパラメーターの値を（値列に）入力し編集します（*上のスクリーンショット*）。パラメータ値の一部の編集は許可されていない場合があることに注意してください。パラメーターの使用を希望しない場合、有効化チェックボックスのチェックを解除して無効化します。この値のセルは編集できないように無効化されます。パラメーターの一部は必須であり、有効化チェックボックスはロックされ、チェックを解除することはできません。パラメーターの値を編集すると、*出力 URI* フィールド内でリクエストが作成されます。パラメーターグリッドには、1つ、または、複数の行がグリッドの下にヘッダーのために含まれています。これらのヘッダーはW ADL ファイルから来ており、W ADL ファイル内の定義に従い定義が許可されている場合、これらの値をグリッド内で編集することができます。ヘッダーのまとめはウィンドウの下のヘッダーフィールドにリストされています。
7. 「OK」をクリックします。HTTP 出力ウィンドウ550 にリクエストがインポートされ、次の方法で表示されます: (i) メソッド リクエストのHTTP メソッドが表示されます。(ii) URL は、リクエストパラメーターをベースにして構成されます。(ii) リクエストのHTTP ヘッダーはヘッダータブに入力されます。(iv) POST または PUT リクエストがリクエストの場合、ボディタブ内にボディが入力されます。
8. 設定を変更する必要があるかを確認するために設定タブをクリックします。
9. 「送信」をクリックしてリクエストを送信します。

(C) 2015-2021 Altova GmbH

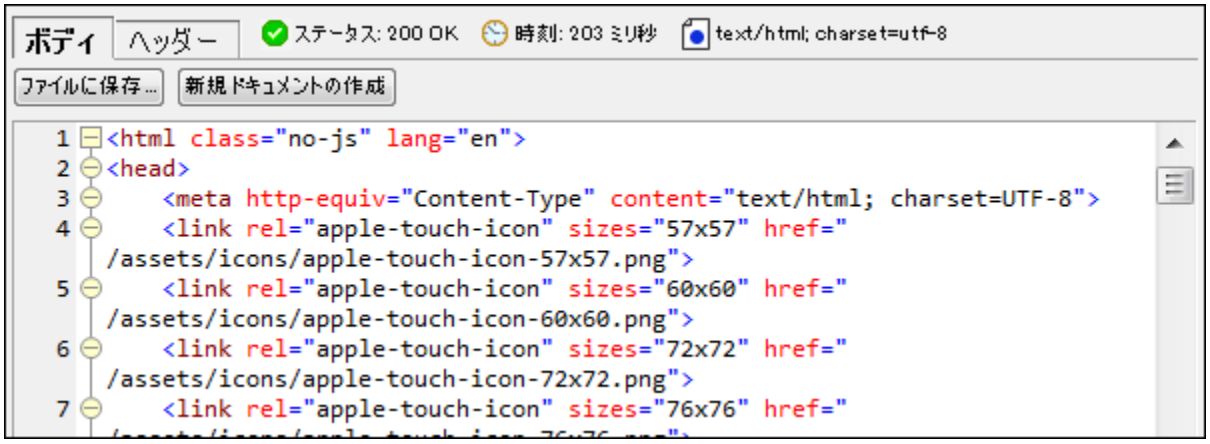
14.3 レスポンスの受信

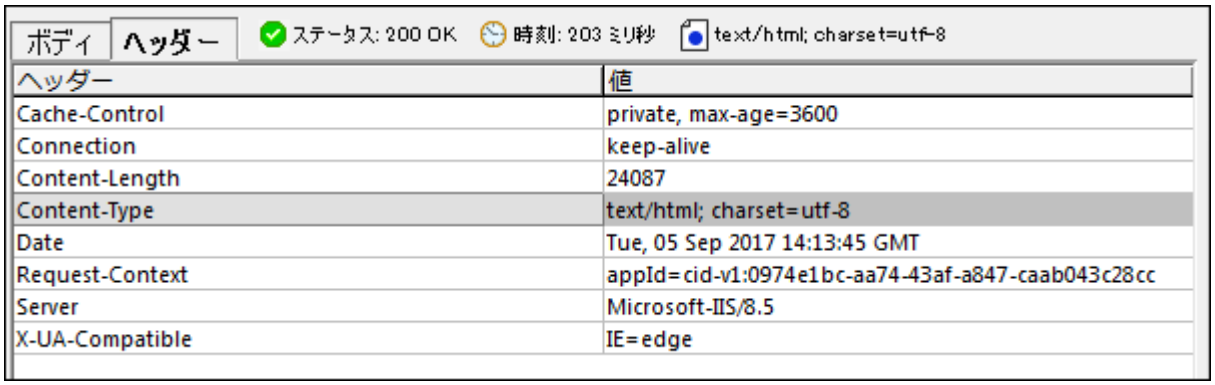
HTTP リクエストへのレスポンスはHTTP 出力ウィンドウの右側のペインで受信されます（*下のスクリーンショットを参照*）。

ボディとヘッダータブ名の右側には、レスポンスに関する以下の詳細がリストされています:

- （下で説明されている558）HTTP 状態コード
- 接続の確立時から最後のレスポンスチャンクが受信された時間
- レスポンスのコンテンツの型

レスポンスペインには2 つのタブが存在します: *ボディ*（*左下のスクリーンショット*）とヘッダー（*右下のスクリーンショット*）.





- *ボディタブ*: ドキュメントがHTML, XML, or JSON の場合、レスポンスのボディが構文色分けと行番号を用いて表示されます。タブ内には2つのボタンがあります: (i) ファイルにボディを保存する「**ファイルに保存する**」。(ii) XMLSpy 内で新規ドキュメントを表示する「**新規ドキュメントを作成する**」とGUI のマインウィンドウ内で新規に作成されたドキュメントが表示されます。新規に作成されたドキュメントは通常の方法で編集し、保存することができます。レスポンスのボディからレスポンスを作成することの出来ない場合、(例えばボディがイメージの場合など)「**新規ドキュメントを作成する**」ボタンが表示されます(*スクリーンショットを参照*)。
- *ヘッダータブ*: レスポンスのヘッダーが含まれています。コンテンツの型 ヘッダーはペインの上の部分に表示されます。

HTTP 状態コード

状態コードは次のように分類されます:

- 2XX は、成功したリクエストのために使用されます。
- 3XX は、リダイレクトのために使用されます。
- 4XX は、リクエストに問題がある場合使用されます。
- 5XX は、サーバーに問題がある場合使用されます。

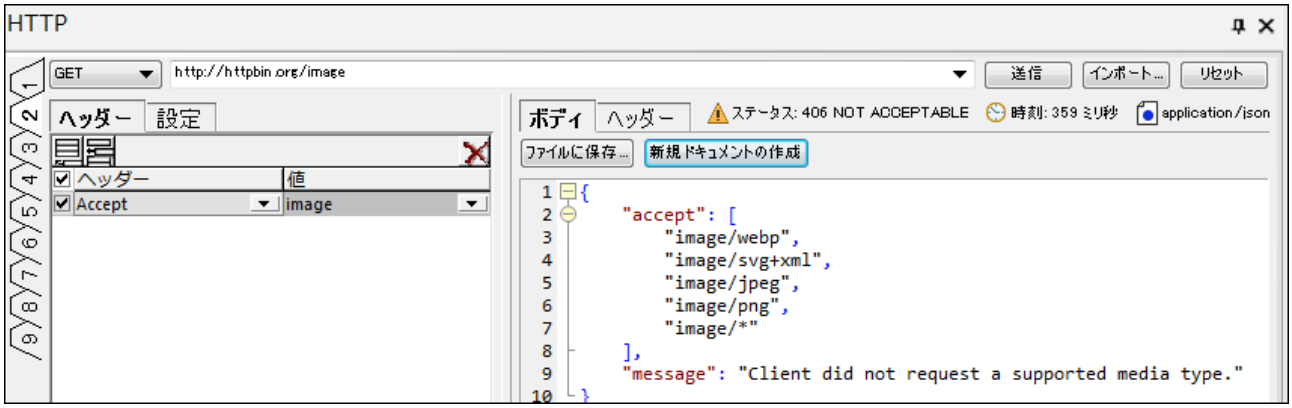
一般に使用されるコードの一部:

- *200 OK*: リクエストの送信に成功した場合。
- *206 部分的内容*: リソースの一部のみがリクエストされているため、サーバーはリソースの一部を返します。
- *301 恒久的に移動した* 与えられたURL にリクエストがリダイレクトされます。
- *401 認証が必要である* リソースは認証を必要とし、認証が失敗したか、または、認証が与えられていません。
- *403 禁止されている* 有効なリクエストですが、サーバーがアクションを拒否しています。これは、ユーザーが必要な資格を有していない可能性を意味します。
- *404 未検出*: リソースが見つからなかった場合。
- *500 サーバー内部エラー*: 詳細なメッセージが存在しない場合送信される、ジェネリックなエラーメッセージ。

次も参照: HTTP 状態コードの完全なリスト。

ヘッダーの受け入れ

リクエストのヘッダーの受け入れは、レスポンス内で受け入れられるコンテンツの型を指定します。例えば、`Accept=image` (*下の最初のスクリーンショット*) と`Accept=image/png` (*下の2番目のスクリーンショット*)の違いを確認してください。最初の場合、レスポンスに送信されるイメージの書式が指定されていないため、エラーの状態コードと詳細を含むJSON メッセージが送信されます。



HTTP
GET
http://httpbin.org/image
送信
インポート...
リセット
ヘッダー
設定
ヘッダー
値
Accept
image/png
ボディ
ヘッダー
ステータス: 200 OK
時刻: 359 ミリ秒
image/png
ファイルに保存...
新規ドキュメントの作成

15 Office Open XML、ZIP、EPUB

Office Open XML (OOXML) やZIP ファイル、そしてEPUB ファイルは、他のファイルを内部に含むパッケージという意味では、すべて似たようなものです。XMLSpy のArchive View では、これらパッケージの内部構造を閲覧するためのビューが提供され、XMLSpy 内部から構造の修正や、パッケージ内のファイルにアクセス、編集を行うことができます。EPUB ファイルの場合、XMLSpy のArchive View からブラウザービュー内にてEPUB ブックを直接閲覧することができます。

Office Open XML (OOXML)

OOXML はドキュメント、スプレッドシート、そしてプレゼンテーションの記述を行うためのファイル形式です。元々はMicrosoft のOffice パッケージのために開発されましたが、現在はオープンなECMA 仕様になっています。

OOXML ファイルの構造

各 OOXML ドキュメントは、Open Package Convention に従った複数ファイルのパッケージになっています。パッケージはXML やその他のデータファイル(例えばイメージファイル)と、それらファイルの関係が記述されたリレーションシップファイルから構成されています。

OOXML ファイルの内部構造や、内部フォルダー、ファイル名の構造はドキュメントの種類により変化します。しかし、`[Content_Types].xml` という共通のXML 構造ファイルがルート部にあり、ドキュメントの種類により変化するディレクトリと、`_rels`、そして `docProps` の3つのディレクトリは常に存在します。例えば、`.docx` ドキュメントの場合、ディレクトリの名前は `word` になり、`.xlsx` ドキュメントの場合は、`xl`、そして `.pptx` の場合は `ppt` となります。

```
OOXML ファイル
| -- ファイル:        [Content_Types].xml
| -- フォルダー:    _rels
| -- フォルダー:    docProps
| -- フォルダー:    word/xl/ppt
```

- `_rels` フォルダーには `rels.xml` ファイルが含まれており、パッケージ内の様々なファイルが記述されています。
- `docProps` フォルダーにはドキュメントプロパティの記述を行う `app.xml` と `core.xml` が含まれています。
- `word`、`xl`、そして `ppt` フォルダーには、ドキュメントのコンテンツとなるXML ファイルが含まれています。例えば、`word` フォルダーには、ドキュメントのコアコンテンツとなる `document.xml` というファイルが保存されています。

XMLSpy のアーカイブビューで見るOOXML

(以下のスクリーンショットにある)XMLSpy のアーカイブビューを使うことで、OOXML ファイルのコンテンツの閲覧と編集を行うことができます。

画面の左側にあるフォルダービューには、パッケージ内のフォルダーが表示され、メインウィンドウにはフォルダービューで選択されているフォルダー内にあるファイルが表示されます。アーカイブビューを使えば、アーカイブからファイルやフォルダーの追加や削除を行うことができます。更に、アーカイブビューのファイルをダブルクリックしてXMLSpy にて素早く編集を行うこともできます。

(C) 2015-2021 Altova GmbH

OOXML の内部ファイルに対するインテリジェントな編集

OOXML パッケージ内部にある XML ドキュメントは、標準的なスキーマをベースにしています。XMLSpy は OOXML ドキュメントに対してインテリジェントな編集機能をサポートしており、入力ヘルパーや自動補完、検証などの機能を使用することができます。

ZIP ファイル

ZIP ファイルは複数のファイルを可逆圧縮の形で含むアーカイブファイルです。含まれるファイルの形式には様々なものがあります。XMLSpy のアーカイブビューを使えば、ZIP ファイルの作成、内部構造の修正や、アーカイブされたファイルの編集を行うことができます。これらの機能については、このセクション以下の ZIP ファイル 566 サブセクションにて記述されます。

EPUB ファイル

EPUB ファイルとはデジタル形式の出版物（EPUB books）を配布するためのファイルの集合体のことです。Archive View 225 では EPUB ファイルを開いたり、EPUB ファイルの作成、デジタル EPUB ブックのプレビュー、EPUB アーカイブのコンポーネントファイルを XMLSpy にて直接編集し、EPUB ファイルを検証、コンポーネントを EPUB アーカイブへ保存しなおすことができます。詳細については EPUB ファイル 568 のセクションを参照ください。

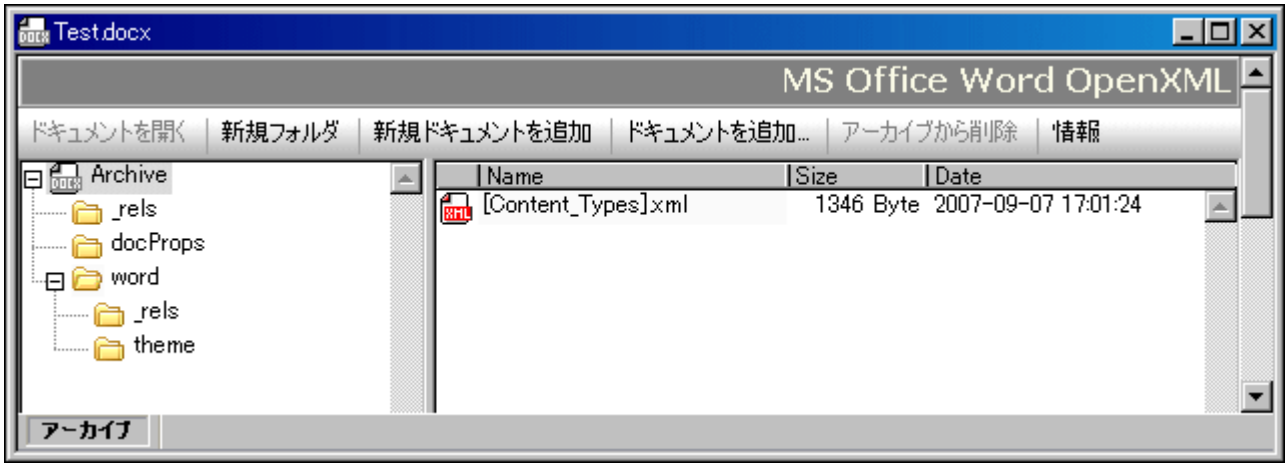
15.1 OOXML ファイルの編集

このセクションでは、Archive View を使ったOOXML ドキュメントの編集方法について説明します。以下の機能について解説します:

- OOXML ファイルを作成、開き、保存する 562
- OOXML ファイルの構造を編集する 562
- OOXML 内部のドキュメントを開き、編集、保存する 563
- OOXML 内部ドキュメントのインテリジェントな編集 563
- OOXML ファイル内部のアドレス 563
- OOXML アーカイブの比較 562

OOXML ファイルを作成、開き、保存する

OOXML ファイルは(「**ファイル | 新規作成**」コマンドから開始できる)新規ドキュメントを作成ダイアログから、目的のファイルの型(`.docx`、`.pptx`、`.xlsx`)を選択して、作成することができます。ファイル名とファイルを保存する場所を指定するよう促されます。指定された場所に新しいファイルが作成され、(以下のスクリーンショットにある)Archive View にて開かれます。OOXML ドキュメントの基本的な内部構造が既に作成されていることに注目してください。



XMLSpy のファイルを開くダイアログ(「**ファイル | 開く**」)により既存のOOXML ファイルを開くこともできます。OOXML の保存を行うには「**ファイル | 上書き保存**」コマンドを使用します。このコマンドによりOOXML ファイルの構造とリレーションシップが保存されます。

OOXML ファイルの構造を編集する

Archive View 225 の機能を使用することでOOXML ファイルのコンテンツにフォルダーやドキュメントの追加、削除を行うことができます。これら構造上の修正を終えると、それらの変更を反映させるために「**ファイル | 上書き保存**」コマンドを使ってOOXML ファイルの保存を行う必要があります。以下の点に注意してください:

- 新規フォルダーまたは新規ドキュメントがArchive View のコマンドボタンから 225 追加された場合、作成時(追加時)にその名前を決定しなければなりません。Archive View でフォルダーまたはドキュメントの名前を変更することはできません。
- 新規ドキュメントがアーカイブフォルダーに追加されると、そのドキュメントを保存するか、OOXML ファイルそのものを保存することで、そのドキュメントが実際にアーカイブに保存されます。

OOXML 内部のドキュメントを開き、編集、保存する

Archive View で表示されている OOXML 内部のドキュメントをダブルクリックするか、メインウィンドウで選択し、ドキュメントを開く 225 コマンドボタンをクリックすることでドキュメントを開くことができます。ドキュメントは XMLSpy のウィンドウで別に開かれます。ドキュメントの編集後は、ドキュメントを保存することで、ドキュメントが OOXML アーカイブファイル内に保存されます。

OOXML 内部ドキュメントのインテリジェントな編集

XMLSpy のインテリジェントな編集機能を OOXML ファイル パッケージ内の OOXML ドキュメントに対して使用することができます。インテリジェントな編集機能により提供される入力ヘルパー、自動補完、そして検証機能を使用することができます。

OOXML ファイル内部のアドレス

OOXML ファイル内のドキュメントのアドレスは、通常のファイル パスとパイプ文字（|）を組み合わせることで表現することができます。例えば、以下のファイル パスは:

```
C:¥Documents and Settings¥<ユーザー名>¥マイ ドキュメント
¥Altova¥XMLSpy2021¥Examples¥Office20XX¥ExcelDemo.xlsx|zip¥xl¥テーブル¥テーブル1.xml
```

XMLSpy サンプルフォルダーの `Examples/Office20XX` フォルダーにある `ExcelDemo.xlsx` OOXML ファイル内に格納されている `xl¥テーブル` フォルダーの `テーブル1.xml` というファイルを表します。

OOXML アーカイブの比較

OOXML ファイルが Archive View で開かれている場合、「ツール | ディレクトリを比較」 987 コマンドを使用することで、他のアーカイブとの比較を行うことができます。

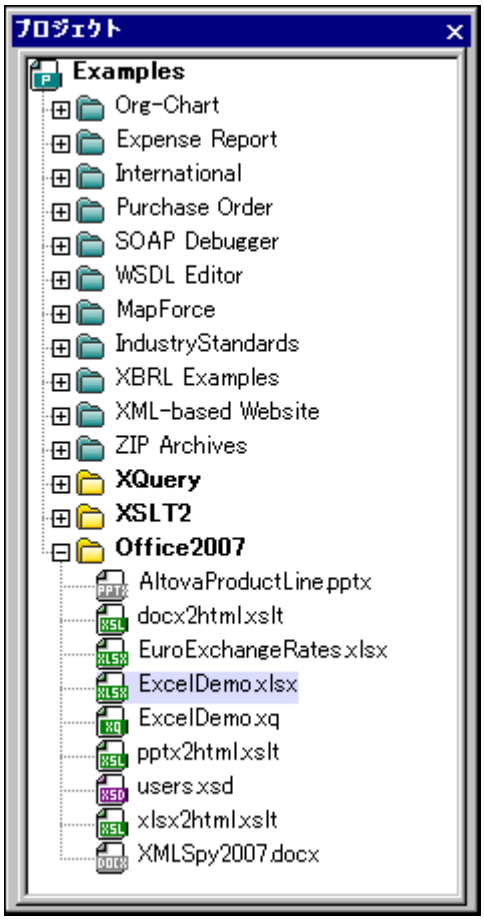
15.2 OOXML サンプルファイル

お使いのXMLSpy アプリケーションフォルダー以下にある Examples\Office フォルダーには、以下のサンプルファイルが配置されています:

- OOXML ファイル: (i) W ord Open XML ファイル(.docx)、(ii) Excel Open XML ファイル(.xlsx)、(iii) PowerPoint Open XML ファイル(.pptx)
- XSLT ファイル: (i) docx2html.xslt (サンプル .docx ファイルを HTML へ変換)、(ii) xslx2html.xslt (サンプル .xslx ファイルを HTML へ変換)、(iii) pptx2html.xslt (サンプル .pptx ファイルを HTML へ変換)
- XQuery ファイル: ExcelDemo.xq (.xslx ファイルからデータを取得する)

XSLT ならびにXQuery ファイルはOOXML ファイル内部のデータにアクセス、変換する際にXSLT やXQuery がどのように使用できるかのデモを行う目的で作成されています。XSLT やXQuery ドキュメントを実行するには、以下のどれかの方法を使用します:

- Archive View にてOOXML ファイルを開きます。フォルダービューにて、Archive を選択し、メニューコマンドから「XSL/XQuery | XSL 変換」を選択するか、「XSL/XQuery | XQuery 実行」を選択します。XSLT またはXQuery ファイルを参照し、「OK」をクリックします。
- XMLSpy のプロジェクトウィンドウにて、(以下のスクリーンショットにあるように) Examples プロジェクトのOffice2007 フォルダー以下にある .xlsx、.pptx または .docx ファイルを右クリックし、変換コマンドを選択します。変換を行うファイルを選択し、「OK」をクリックします。



- XMLSpy にて XSLT または XQuery ファイルを開き、メニューコマンドから「**XSL/XQuery | XSL 変換**」または「**XSL/XQuery | XQuery 実行**」をそれぞれ選択します。変換を行う XML を選択するよう促されるので、`.docx`、`.xlsx`、または `.pptx` ファイルを選択します。
- これらのオペレーションは Altova XSLT ならびに XQuery エンジンを搭載したスタンドアロンパッケージの AltovaXML を使っても行うことができます。

15.3 ZIP ファイル

Archive View[225] では ZIP ファイルを作成し、ファイル内部の構造を修正し、XMLSpy にて ZIP パッケージに含まれるファイルを直接編集し、そのファイルを ZIP アーカイブへ保存しなおすことができます。

ZIP ファイルの作成と保存

ZIP ファイルは新規ドキュメント作成ダイアログ(「**ファイル | 新規作成**」コマンド)にて `.zip` を選択することで作成することができます。(スクリーンショットにあるような)空の ZIP アーカイブが XMLSpy で作成されます。その後「**ファイル | 上書き保存**」コマンドを使って ZIP ファイルを適切な場所に保存する必要があります。以下に記されるようにファイルやフォルダーを追加して、これらの変更を保存するために ZIP ファイルを保存します。

既存の ZIP ファイルは XMLSpy のファイルを開くダイアログ(「**ファイル | 開く**」)から開くことができます。

注意: OOXML ファイルの作成時には、アーカイブが Archive View で開かれる前に保存場所を尋ねられるのに対して、ZIP ファイルの場合、作成時に保存場所を選択するよう促されることはありません。Archive View で開かれている空のアーカイブの ZIP ファイルを保存するには、明示的に「**ファイル | 上書き保存**」コマンドを選択する必要があります。

フォルダーやファイルの追加とアーカイブ構造の修正

選択されたアーカイブフォルダーに対してフォルダー(**新規フォルダー**)や、既存のドキュメント(**ドキュメントを追加**)、そして新規ファイル(**新規ドキュメントを追加**)を追加することができます。新規フォルダーや新規ドキュメントを追加したときには、直ちにフォルダーまたはファイルの名前を入力する必要があります。Archive View にてフォルダーやファイルの名前の変更を行うことはできません。

ZIP ファイル内部のアドレス

ZIP ファイル内部のドキュメントは、通常のパスとパイプ文字(|)を組み合わせることで表現することができます。例えば、以下のファイルパスは:

```
C:¥Documents and Settings¥<ユーザー名>¥マイ ドキュメント¥Altova¥XMLSpy2021¥Examples¥Test.zip|zip\TestFolder\MyFile.xml
```

XMLSpy サンプルフォルダーの `Examples` フォルダーにある `Test.zip` ZIP ファイル内に格納されている `TestFolder` フォルダーの `MyFile.xml` というファイルを表します。

ZIP アーカイブの比較

ZIP ファイルが Archive View で開かれている場合、「ツール | ディレクトリの比較」[987] コマンドを使用することで、別のアーカイブとの比較を行うことができます。

15.4 EPUB ファイル

EPUB ファイルとはデジタル形式の出版物（EPUB books）を配布するためのファイルの集合体のことです。Archive View[225] では EPUB ファイルを開き、EPUB ファイルの作成、デジタル EPUB ブックのプレビュー、EPUB アーカイブのコンポーネントファイルを XMLSpy にて直接編集し、EPUB ファイルを検証、コンポーネントを EPUB アーカイブへ保存し直すことができます。詳細については EPUB ファイル[568] のセクションを参照ください。

メモ: **(i)** XMLSpy では EPUB 2.0.1 がサポートされ、**(ii)** サンプル EPUB ファイルは `(My) Documents/Altova/XMLSpy2021/Examples` フォルダー内にある `Examples` プロジェクトにてご利用になれます。

用語解説

このセクションでは、以下のような用語が使用されます:

- **EPUB ファイル**とは、`.epub` というファイル拡張子をともなう EPUB ファイルのことです。ファイルそのものは ZIP ファイルで、Archive View にて開くことができる全てのファイルがアーカイブ内に含まれます。
- **アーカイブファイル**とは、EPUB アーカイブを含んでいるファイルのことです。
- **EPUB ブック**という用語は、アーカイブされた EPUB ファイルにより生成されたデジタル形式の出版物のことを指します。

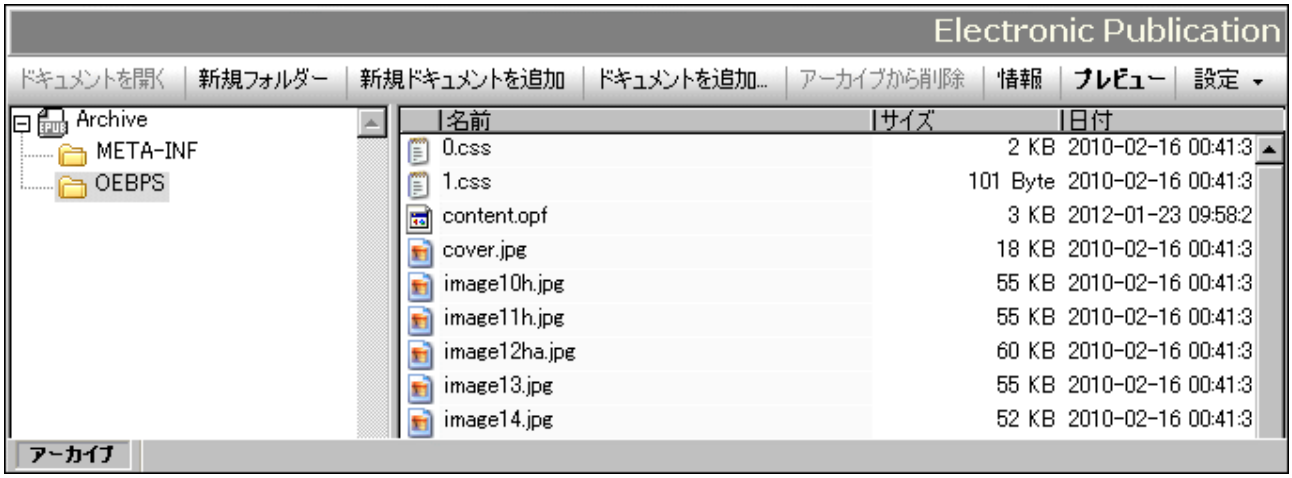
このセクションの構成

このセクションでは、XMLSpy における EPUB の機能が以下のように構成されます:

- Archive View にて EPUB ファイルを開く[568]
- 新規 EPUB ファイルを作成する[569]
- EPUB ブックのプレビュー[569]
- EPUB アーカイブのコンテンツや構造を修正する[570]
- 情報と設定[570]
- XMLSpy にてアーカイブファイルを編集する[570]
- アーカイブファイルの入力ヘルパー[570]
- EPUB ファイルの検証[570]

Archive View にて EPUB ファイルを開く

メニューコマンドから「**ファイル | 開く**」を選択し、EPUB ファイルを選択します。EPUB ファイルが Archive View にて表示されます（以下のスクリーンショットを参照）。Windows エクスプローラーにて EPUB ファイルを右クリックし、コンテキストメニューからファイルを XMLSpy にて開くこともできます。XMLSpy を EPUB ファイルのデフォルトエディターとすることで[1018]、EPUB ファイルをダブルクリックすることで、ファイルが Archive View にて開かれるようになります。



左側にはアーカイブ内にあるフォルダーが表示され、メインウィンドウには、フォルダービューにて選択されたフォルダー内にあるファイルが表示されます。EPUB アーカイブには以下のような構造とコンポーネントが表示されます。

```
Archive
|-- Mimetype ファイル
|
|-- META-INF フォルダー
|     |-- container.xml
|
|-- ドキュメントフォルダー（上のスクリーンショットでは OEBPS がドキュメントフォルダーとなります）
      |-- HTML、CSS、イメージファイル、そして OPF ならびに NCX ファイルが含まれます
```

新規 EPUB ファイルを作成する

新規にEPUB ファイルを作成するには、メニューコマンドから「**ファイル | 新規作成**」を選択します。新規ドキュメントの作成ダイアログにて、.epub というファイルの型を選択します。名前を付けて保存ダイアログにて、EPUB ドキュメントの名前を入力し、「**保存**」ボタンをクリックします。妥当な EPUB アーカイブを構成するために必要な全てのファイルが含まれるスケルトンのEPUB アーカイブが、Archive View 内の新規ウィンドウにて作成されます。以下に記される方法により、必要に応じてファイルやフォルダーをアーカイブへ追加して、EPUB ファイルを保存します。XMLSpy にて直接アーカイブファイルを編集するには、Archive View にあるファイルをダブルクリックします。新たな XMLSpy ウィンドウにてファイルが開かれます。ファイルの編集を行い、「**ファイル | 保存**」コマンドにより保存してください。

EPUB ブックのプレビュー

EPUB ブックをプレビューするには、Archive View にて EPUB ファイルがアクティブになっている状態で、Archive View のツールバーにてプレビューボタンをクリックします。EPUB ブックがXMLSpy 内にある（Internet Explorer）ブラウザーウィンドウにて開かれます。これらファイルのどれか-コンテンツファイルであれ構造に関連するファイルであれ- が変更され、まだ保存されていない場合、ファイルを保存するように促されます。変更点を保存しない場合、以前保存されたデータがプレビューには使用されます。Archive View のツールバーにある「**設定**」ボタンから、プレビューを行う前に、変更されたファイルを保存するように指定することができます。

以下の点に留意してください:

- EPUB データがプレビューウィンドウで開かれている時にアーカイブビューの「**プレビュー**」ボタンをクリックすると、EPUB データが既に開かれているプレビューウィンドウで再ロードされます。
- プレビューウィンドウ（Internet Explorer の再描画（F5）コマンドにより）を再描画しても、プレビューウィンドウそのものは更新されません。アーカイブビューの「**プレビュー**」ボタンを使用して、プレビューウィンドウ内のEPUB データを更新してください。
- プレビューウィンドウを閉じることで、プレビューを閉じることができます。

メモ: 全てのEPUB マークアップがInternet Explorer によりサポートされているわけではなく、プレビューのレイアウトが崩れることもあります。更に、デジタル出版物のドキュメントがHTML ではなくXML をベースにしたものである場合、プレビューが正常に行われない可能性もあります。最新バージョンのInternet Explorer ではEPUB マークアップの扱いが改善されており、問題が発生した場合は、最新バージョンのInternet Explorer へ更新することをおすすめします。

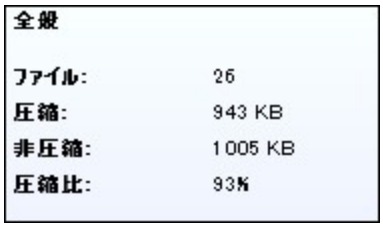
EPUB アーカイブのコンテンツや構造を修正する

選択されたアーカイブフォルダーに対して、フォルダーを追加(「**新規フォルダー**」ボタンをクリック)、新規ファイルを追加(「**新規ドキュメントの追加**」ボタン)、既存のファイルを追加(「**ドキュメントの追加**」ボタン)することができます。新規フォルダーや新規ドキュメントを追加する場合、そのファイルやフォルダーの名前を直ちに入力する必要があり、Archive View にてフォルダー名を変更することはできません。ファイルやフォルダーを選択して、「**アーカイブから削除**」ボタンをクリックすることで、ファイルやフォルダーを削除することができます。

アーカイブを変更したら、それら変更を反映させるために(「**ファイル | 保存**」コマンドにて)EPUB ファイルを保存する必要があります。

情報と設定

「**情報**」ボタンをクリックすることで、Archive View の下部にアーカイブ情報の概要が表示されます(以下のスクリーンショットを参照)。再度「**情報**」ボタンをクリックすることで、概要が非表示となります。概要には、アーカイブ内にある(Mimetype や `container.xml` を含む)ファイルの数や圧縮されたEPUB ファイルの大きさ、圧縮されていない状態のファイルの大きさが示されます。



「**設定**」ボタンをクリックすることで、ファイルの自動保存オプションに対応したドロップダウンが表示されます。このコマンドにより、(i) 検証前、そしてEPUB ファイルを(XMLSpy の「**プレビュー**」ボタンにより) プレビューする前にファイルを自動保存するように指定することができます。

XMLSpy にてアーカイブファイルを編集する

アーカイブファイルをXMLSpy にて直接編集するには、Archive View に表示されているファイルをダブルクリックします。または、Archive View にて目的のファイルを選択し、Archive View のツールバーからドキュメントを開くボタンをクリックします。XMLSpy にてファイルが開かれます。ファイルを編集した後に、「**ファイル | 保存**」コマンドにてファイルを保存するようにしてください。

アーカイブファイルの入力ヘルパー

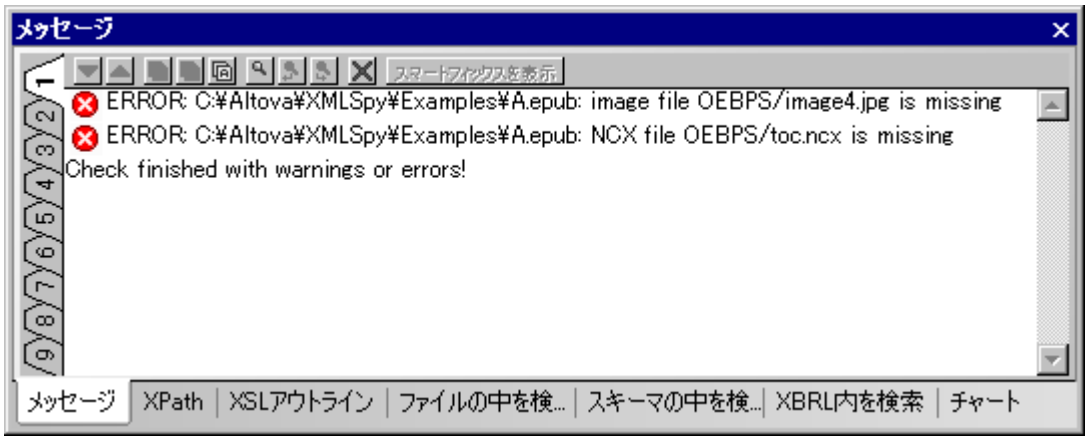
アーカイブファイルがXMLSpy にて開かれると、標準的なアーカイブファイルに対応した入力ヘルパーが表示されます。アーカイブファイルは以下のようなものになります:

- 通常 `content.opf` という名前のOPF ファイルにはEPUB ブックのメタデータが含まれます。Open Packaging Format (OPF) の仕様をベースにしています。
- 通常 `toc.ncx` という名前のNCX ファイル(XML に対するナビゲーションコントロールファイル)には、出版物の目次が含まれます。OPF 仕様のNCX パートをベースにしています。
- `META-INF` という名前のフォルダーには `container.xml` という名前のファイルが含まれており、ブック(OPF ファイル)のコンテンツを定義するファイルを指し示します。`container.xml` ファイルによりOpen Container Format (OCF) 仕様のルールに従ったかたちでアーカイブファイルが構成されます。

EPUB ファイルの検証

EPUB ファイルを検証するには、「**XML | XML の検証**」コマンドを選択します。検証結果がメッセージウィンドウに表示されます(以下のスクリーンショットを参照)。アクティブなファイルのどれか-コンテンツファイルであれ構造に関連するファイルであれ- が変更され、まだ保存されていない場合、ファイルを保存するように促されます。EPUB ファイルの検証を行うには、変更されたファイルを保存する必要があります。

Archive View のツールバーにある「設定」ボタンから、プレビューを行う前に、変更されたファイルを保存するように指定することができます。



エラーメッセージには、(i) エラーが発見されたファイルの場所と、可能な場合はエラーが発生した行番号、(ii) エラーの詳細が表示されます。上のスクリーンショットでは、`content.opf` ファイルの21行目にてエラーが発生したことが示されています。メッセージウィンドウ内のエラー行をクリックすることで関連するファイルが開かれ、エラーの箇所がハイライトされます。

メモ: EPUB 検証エンジンは Java のユーティリティーで、検証エンジンを使用するにはお使いのコンピューターに Java がインストールされていなければなりません。

16 データベース

XMLSpy を使えば様々なデータベース(DB) へ接続し、DB に対して問い合わせを行ったり、DB の構造を XML スキーマとしてインポート、DB から XML データファイルを生成し、DB へデータをエクスポートすることができます。DB に関連した各機能は XMLSpy のメニューコマンドから扱うことができ、詳細はユーザーマニュアル[741]に記述されています。これらのコマンドの完全なリストを、対応する記述へのリンクと共に以下に記します。

このセクションは、以下のように構成されます:

- データベースへの接続方法[573]: XMLSpy の DB に関するコマンドを実行する際に必要なオペレーションを説明します。
- テストにより XMLSpy で既に使用できることが確認されている DB のリストを示します[659]。

注意: 64ビット版の XMLSpy をお使いの場合は、DB への接続に使用されるデータベースドライバーも64ビットである必要があります。

XMLSpy の DB に関する機能

XMLSpy の DB に関する機能は「DB」[912] ならびに「変換」[938] メニューコマンドから利用できます。

- *データベースクエリの実行*[912]: 「DB」メニューに存在します。DB の構造を別のデータベースクエリウィンドウにロードし、DB への問い合わせを行ないます。結果はデータベースクエリウィンドウに表示されます。
- *IBM DB2*[925]: 「DB」メニューに存在します。DB2 は XML DB で、XMLSpy により XML DB の XML スキーマ管理や、XML DB の編集ならびに検証を行うことができます。
- *SQL Server*[931]: 「DB」メニューに存在します。XMLSpy は DB の XML スキーマと編集、検証機能の管理を有効にします。
- *Oracle XML DB*[933]: 「DB」メニューに存在します。XML スキーマ管理、データベースの問い合わせ、そして DB スキーマをベースにした XML ファイルの生成など、Oracle XML DB に対する数多くの機能を提供します。
- *データベースのデータをインポート*[941]: 「変換」メニュー内に存在します。DB データを XML ファイルへインポートします。
- *データベース構造から XML スキーマを作成*[946]: 「変換」メニュー内に存在します。DB の構造をベースにした XML スキーマを生成します。
- *XML スキーマを元にデータベースをインポート*[950]: 「変換」メニュー内に存在します。XMLSpy にて XML スキーマドキュメントがアクティブになっている状態で、DB の接続を確立し、選択された DB テーブルのデータをインポートします。その結果作成される XML ドキュメントは DB 接続を確立したときにアクティブになっている XML スキーマをベースとします。
- *XML スキーマを元にデータベース構造を作成*[951]: 「変換」メニュー内に存在します。既存の XML スキーマの構造をベースにした、データを持たない DB テーブルが作成されます。
- *データベースへエクスポート*[957]: 「変換」メニュー内に存在します。XML ドキュメントのデータを DB へエクスポートします。XML データを使って既存の DB テーブルを更新するか、XML データを含んだ新しいテーブルを作成することができます。

データ型の変換

XML ドキュメントと DB 間のデータ変換を行うにあたり、データの型も対応する形式に変換する必要があります。XMLSpy によるデータ型の変換方法は付録の DB により生成された XML のデータ型[1578] と XML スキーマから生成された DB のデータ型[1584] に収められています。

Altova DatabaseSpy

Altova の DatabaseSpy は複数のデータベースに対応した DB デザインツールで、XMLSpy にはない機能が提供されます。Altova DatabaseSpy に関する詳しい情報については、Altova ウェブサイトをご覧ください。

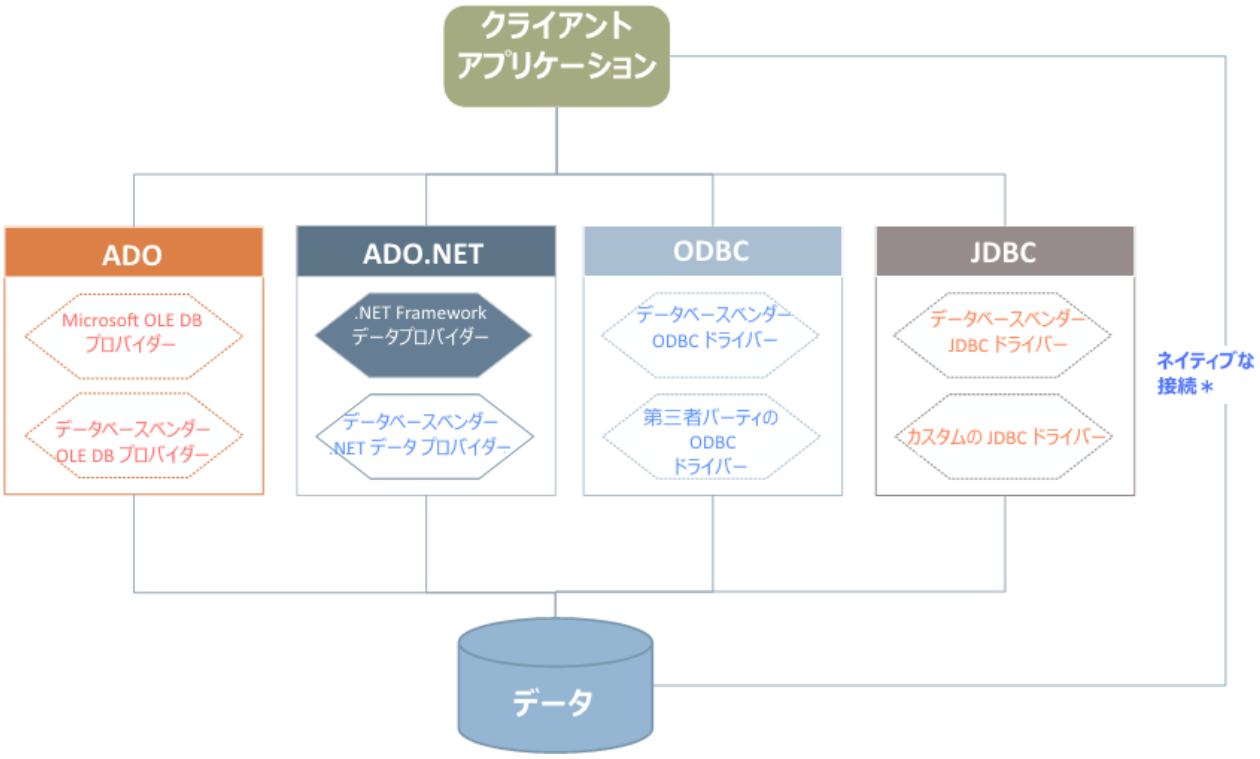
(C) 2015-2021 Altova GmbH

16.1　データソースへの接続

最も簡単なケースとして、データベースは、Microsoft Access、または、SQLite データベースファイルなどのローカルファイルが挙げられます。高度なケースの例としては、データベースは、アプリケーションが接続し、データを消費するオペレーティングシステムと同じオペレーティングシステムを使用するとは限らない、リモート、または、ネットワークデータベースサーバーに存在します。例えば、XMLSpy が、Windows で作動し、データ（例えばMySQL）へアクセスするデータベースがLinux マシーンで作動することは可能です。

多種のデータベース型とリモートおよびローカルで作業するには、XMLSpy は、データ接続インターフェイスとオペレーティングシステム内で使用することのできる、または、定期的にメジャーなデータベースベンダーによりリリースされるデータベースドライバーに従います。常に進化するデータベース技術の景観の中、このアプローチは、クロスプラットフォームに対する柔軟性および相互運用性に対応します。すなわち、XMLSpy により、以下のデータアクセス技術により主要なデータベースにアクセスすることができます

次の図は、最も簡単な方法で、（ジェネリックなクライアントアプリケーションとして表示されている）XMLSpy と（データベースサーバー、または、データベースファイルであることができる）データ保管の間で使用することのできるデータ接続のオプションを示しています。



＊直接なネイティブな接続 は、SQLite とPostgreSQL データベースのためにサポートされています。使用中のシステムにインストールするために追加ドライバーは必要ではありません。

上の図で示されるように、XMLSpy は、主なデータベース型に次のアクセス技術を使用してアクセスすることができます:

- 基となるOLE DB（Object Linking and Embedding, Database）プロバイダーを使用するADO（Microsoft® ActiveX® Data Objects)
- ADO.NET（データとのインタアクションを有効化するMicrosoft .NET Framework 内で使用することのできるライブラリのセット）
- JDBC（Java データベース接続）
- ODBC（Open データベース接続）

ADO.NET の一部はサポートされていない、または、サポートが制限されている場合があります。次を参照してください: ADO.NET サポートメモ[590]。

選択するデータベースへの接続のインターフェイスは、既存のソフトウェアのインフラストラクチャによります。データのアクセス技術と、接続するデータベースと密に統合することのできる、データベースドライバーを通常選択します。例えば、Microsoft Access 2013 データベースに接続するには、Microsoft Office Access データベース Engine OLE DB プロバイダーなどのネイティブなプロバイダを使用する ADO 接続文字列を作成します。に接続するには、他方、最新の JDBC、ODBC、または ADO.NET インターフェイスを Oracle Web サイトからダウンロードします。

Windows 製品 (Microsoft Access または SQL Server など) のためのドライバーは、Windows オペレーティングシステムで使用可能ですが、他のデータベースの種類では使用することができない可能性があります。メジャーなデータベースベンダーは、対応するデータベースに ADO、ADO.NET、ODBC、または JDBC の組み合わせからクロスプラットフォームアクセスをすることができる、一般使用可能なデータベースクライアントソフトウェアとドライバーを定期的にリリースします。更に、上記の技術のためのサードパーティーのドライバーを使用することができます。多くの場合は、1つ以上の方法で必要なデータベースへ使用中のオペレーティングシステムから接続することができます、この結果、XMLSpy からも接続することができます。使用可能な機能、パフォーマンスパラメーター、と既知の問題はデータアクセス技術および使用されるドライバーにより異なります。

16.1.1 接続ウィザードの開始

XMLSpy にはデータソースをセットアップするために必要とするステップを導くウィザードが搭載されています。

ウィザードのステップを開始する前に、データベースの一部の種類は、データベースドライバーまたはデータベースクライアントソフトウェアなどの複数のデータベースの必須条件を個別にインストールまたは構成する必要があることに注意してください。これらは、通常各データベースのベンダーにより提供され、特定の Windows バージョン用に調整されたドキュメントを含みます。データベースの種類別にグループ化されたデータベースドライバーのリストに関しては、をデータベースドライバーの概要[575]参照してください。

データベース接続ウィザードを開始するには次を行ってください:

- **DB** メニューから、「**データベースのクエリ**」をクリックします。

データベース型を選択し「次へ」をクリックすると データベースの種類、技術（ADO, ADO.NET, ODBC, JDBC）と使用されるドライバーにより異なる命令が画面に表示されます。

例えば、各データベースの種類に適用できるサンプルはデータベース接続のサンプル 603 のセクションを参照してください。各データベースへのアクセス技術に関しては以下のトピックを参照してください：

- ADO 接続の設定 578
- ADO.NET 接続の設定 585
- ODBC 接続の設定 591
- JDBC 接続の設定 594

16.1.2　データベースドライバー概要

以下のテーブルは、特定のデータベースに特定のデータアクセス技術を使用して接続するためによく使用されるデータベースのドライバーのリストです。このリストは、完全または規範的なものではありません。以下に表示されているドライバに加え他のネイティブまたはサードパーティー代替を使用することができます。

Windows オペレーティングシステムの複数のデータベースドライバーが既に使用されている場合がありますが、代替ドライバーをダウンロードする必要があります。データベースの一部に対して、オペレーティングシステムに付随するドライバーよりもデータベースベンダーより提供される最新のドライバーのパフォーマンスの方が良い場合があります。

データベースのベンダーは、個別にダウンロードすることのできるパッケージ、または、バンドルされたデータベースクライアントソフトウェアを提供します。後者の場合、データベースクライアントソフトウェアは、通常、データベースの管理と接続、およびデータベースクライアントとそのコンポーネントのインストールと構成を簡素化するために管理と構成ユーティリティにより構成されています。

データベースクライアントを正確に構成することは、データベースへの接続を成功させるために必須です。データベースをインストールして、使用する前に、各データベースバージョンおよび各 Windows バージョンにより通常異なるデータベースのインストールおよび構成の手順を確認することが奨励されます。

データベースの種類に関して各データアクセステクノロジーの機能と制限を理解するには、特定のデータベース製品のドキュメントを参照し、特定の環境に対しての接続をテストします。よく発生する接続問題を回避するために、以下に注意してください：

- ADO.NET の一部のプロバイダーはサポートされない、または、サポートが制限されています。次を参照してください：ADO.NET サポートメモ[590]。
- データベースドライバーをインストールする際、Altova アプリケーション（32-ビット または 64-ビット）などの同じプラットフォームが使用されることが奨励されます。例えば、32ビットの Altova アプリケーションが 64-ビットオペレーティングシステムで使用される場合、が使用される場合、32-ビットのドライバーをインストールし、32-ビットのドライバーを使用して、データベースの接続をセットアップします。詳細に関しては、使用可能な ODBC ドライバーの表示[593] を参照してください。
- ODBC データソースをセットアップする際、データソース名（DSN）をユーザー *DSN* の代わりに *システム DSN* として作成することが奨励されます。詳細に関しては、使用可能な ODBC ドライバーの表示[593] を参照してください。
- When setting
- JDBC データソースをセットアップする際、JRE（Java Runtime Environment）、Java Development Kit（JDK）がインストールされている、またオペレーティングシステムの CLASSPATH 環境変数が構成されていることを確認してください。詳細に関しては、JDBC 接続のセットアップ[594] を参照してください。
- データベースベンダーからインストールするドライバーおよびデータベースクライアントソフトウェアのインストールの手順とサポートの詳細は、インストールパッケージのドキュメントを参照してください。公式なまたはサードパーティデータベースドライバーの場合、特定のオペレーティングシステムでの特定のドライバーに適用することのできる最も包括的な情報と構成手順は、通常ドライバーインストールパッケージの一部です。

データベース	インターフェイス	ドライバー
Firebird	ADO.NET	Firebird ADO.NET Data Provider (https://www.firebirdsql.org/en/additional-downloads/)
	JDBC	Firebird JDBC driver (https://www.firebirdsql.org/en/jdbc-driver/)
	ODBC	Firebird ODBC driver (https://www.firebirdsql.org/en/odbc-driver/)
IBM DB2	ADO	IBM OLE DB Provider for DB2
	ADO.NET	IBM Data Server Provider for .NET
	JDBC	IBM Data Server Driver for JDBC and SQLJ
	ODBC	IBM DB2 ODBC Driver
IBM DB2 for i	ADO	• IBM DB2 for i5/OS IBMDA400 OLE DB Provider • IBM DB2 for i5/OS IBMDARLA OLE DB Provider • IBM DB2 for i5/OS IBMDASQL OLE DB Provider
	ADO.NET	.NET Framework Data Provider for IBM i
	JDBC	IBM Toolbox for Java JDBC Driver
	ODBC	iSeries Access ODBC Driver
IBM Informix	ADO	IBM Informix OLE DB Provider

データベース	インターフェイス	ドライバー
	JDBC	IBM Informix JDBC Driver
	ODBC	IBM Informix ODBC Driver
Microsoft Access	ADO	• Microsoft Jet OLE DB Provider • Microsoft Access Database Engine OLE DB Provider
	ADO.NET	.NET Framework Data Provider for OLE DB
	ODBC	• Microsoft Access Driver
MariaDB	ADO.NET	MariaDB のための専用の.NET コネクタが不足している場合、MySQL のためのコネクタ/NET を使用してください（https://dev.mysql.com/downloads/connector/net/).
	JDBC	MariaDB Connector/J (https://downloads.mariadb.org/)
	ODBC	MariaDB Connector/ODBC (https://downloads.mariadb.org/)
Microsoft SQL Server	ADO	• Microsoft OLE DB Driver for SQL Server (MSOLEDBSQL) • Microsoft OLE DB Provider for SQL Server (SQLOLEDB) • SQL Server Native Client (SQLNCLI)
	ADO.NET	• .NET Framework Data Provider for SQL Server • .NET Framework Data Provider for OLE DB
	JDBC	• Microsoft JDBC Driver for SQL Server (https://docs.microsoft.com/en-us/sql/connect/jdbc/microsoft-jdbc-driver-for-sql-server)
	ODBC	• ODBC Driver for Microsoft SQL Server (https://docs.microsoft.com/en-us/SQL/connect/odbc/download-odbc-driver-for-sql-server)
MySQL	ADO.NET	• Connector/NET (https://dev.mysql.com/downloads/connector/net/)
	JDBC	Connector/J (https://dev.mysql.com/downloads/connector/j/)
	ODBC	Connector/ODBC (https://dev.mysql.com/downloads/connector/odbc/)
Oracle	ADO	• Oracle Provider for OLE DB • Microsoft OLE DB Provider for Oracle
	ADO.NET	Oracle Data Provider for .NET (http://www.oracle.com/technetwork/topics/dotnet/index-085163.html)
	JDBC	• JDBC Thin Driver • JDBC Oracle Call Interface (OCI) Driver これらのドライバーは、通常、データベースクライアントのインストール中にインストールされます。（Thin ドライバーではない） OCI ドライバーを使用する接続は、Oracle XML DB のコンポーネントを使用しています。
	ODBC	• Microsoft ODBC for Oracle • Oracle ODBC Driver（通常、Oracle データベースクライアントのインストール中にインストールされます）

データベース	インターフェイス	ドライバー
PostgreSQL	JDBC	PostgreSQL JDBC Driver (https://jdbc.postgresql.org/download.html)
	ODBC	psqlODBC (https://odbc.postgresql.org/)
	Native Connection	使用可能。ネイティブな接続を使用する場合、ドライバーをインストールする必要はありません。
Progress OpenEdge	JDBC	JDBC Connector (https://www.progress.com/jdbc/openedge)
	ODBC	ODBC Connector (https://www.progress.com/odbc/openedge)
SQLite	Native Connection	使用可能。ネイティブな接続を使用する場合、ドライバーをインストールする必要はありません。
Sybase	ADO	Sybase ASE OLE DB Provider
	JDBC	jConnect™ for JDBC
	ODBC	Sybase ASE ODBC Driver
Teradata	ADO.NET	.NET Data Provider for Teradata (https://downloads.teradata.com/download/connectivity/net-data-provider-for-teradata)
	JDBC	Teradata JDBC Driver (https://downloads.teradata.com/download/connectivity/jdbc-driver)
	ODBC	Teradata ODBC Driver for Windows (https://downloads.teradata.com/download/connectivity/odbc-driver/windows)

16.1.3 ADO 接続の設定

Microsoft ActiveX Data Objects (ADO) はOLE DB を介して多種のデータソースにデータベースアクセステクノロジーです。OLE DB はODBC またはJDBC への代替インターフェイスです。COM (コンポーネント オブジェクト モデル) 環境内での統一されたアクセスを提供します。ADO は新規のADO.NET 585 前のバージョンであり、他のデータソースのためにも使用可能ですがMicrosoft Access or SQL Server などのMicrosoft ネイティブなデータベースにアクセスを可能にする方法です。

需要な点は、複数のADO プロバイダー間から選択することができ、一部はダウンロードされワークステーションに使用する前にインストールされる必要があります。SQL Server への接続のために以下のADO プロバイダーを使用することができます:

- Microsoft OLE DB *Driver* for SQL Server (MSOLEDBSQL)
- Microsoft OLE DB *Provider* for SQL Server (SQLOLEDB)
- SQL Server Native Client (SQLNCLI)

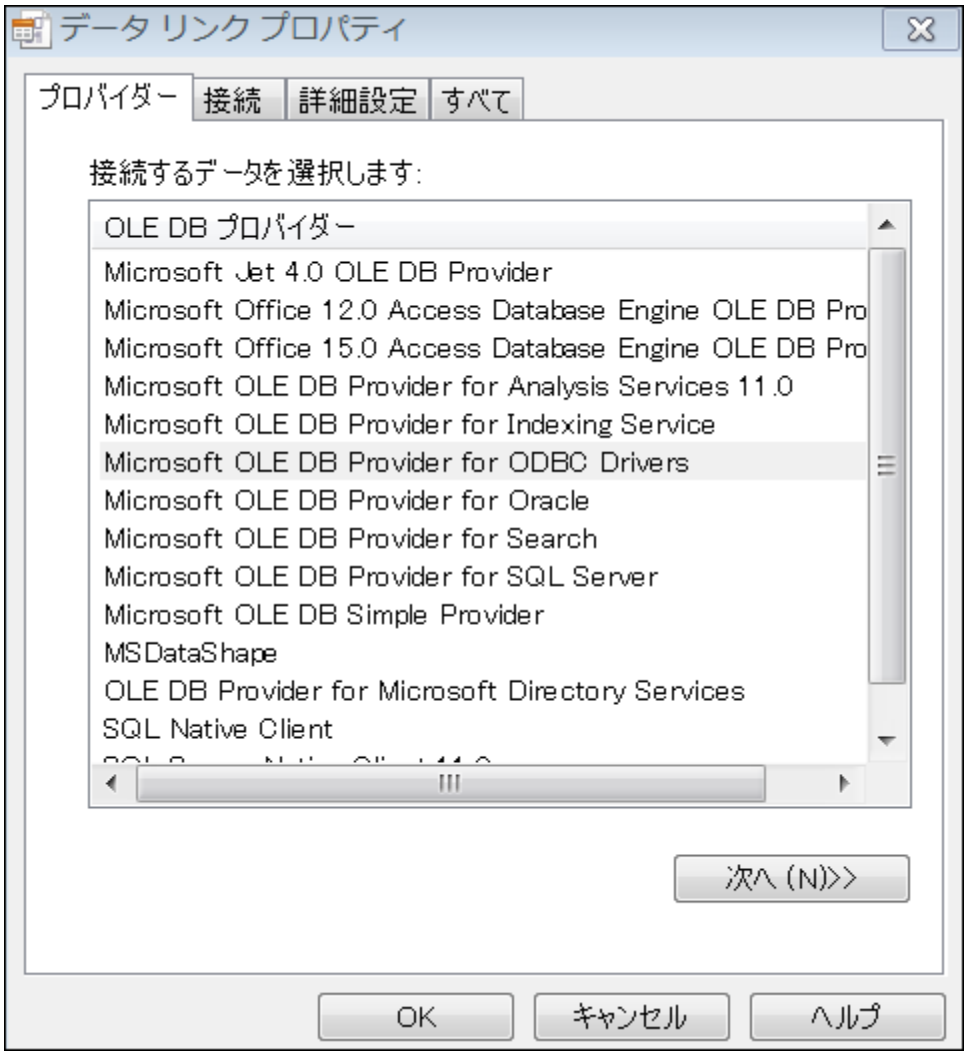
上記でリストされたプロバイダーで奨励されるプロバイダーはMSOLEDBSQL です。https://docs.microsoft.com/en-us/sql/connect/oledb/download-oledb-driver-for-sql-server?view=sql-server-ver15 からダウンロード可能です。XMLSpy (32-ビット または64-ビット)のプラットフォームに一致する必要があることに注意してください。SQLOLEDB とSQLNCLI プロバイダーは非奨励であると考えられています。

Microsoft OLE DB Provider for SQL Server (SQLOLEDB) は共通テーブル式（CTE）とネストされたSELECTステートメントなどの複雑なクエリのパラメーターバインディングに関する問題があることが知られています。

ADO 接続の設定:

1. データベース接続ウィザードを開始します574。
2. 「ADO 接続」をクリックします。

3. 構築] をクリックします。



4. 接続するデータプロバイダーを選択します。下のテーブルは一般的なシナリオをリストしています。

このデータベースに接続する方法...	このプロバイダーを使用する...
Microsoft Access	• **Microsoft Office Access Database Engine OLE DB Provider** (奨励) • **Microsoft Jet OLE DB Provider** **Microsoft Office Access Database Engine OLE DB Provider** がリスト内で使用できない場合 Microsoft Access または Microsoft Access Database Engine Redistributable (https://www.microsoft.com/en-us/download/details.aspx?id=54920) がコンピューターにインストールされていることを確認してください。
SQL Server	• **Microsoft OLE DB Driver for SQL Server (MSOLEDBSQL)** が OLE DB プロバイダーのために奨励されています。リストにこのプロバイダーが表示されるためには、プロバイダーが https://docs.microsoft.com/en-

このデータベースに接続する方法…	このプロバイダーを使用する…
	us/sql/connect/oledb/download-oledb-driver-for-sql-server?view=sql-server-ver15 からダウンロードされインストールされる必要があります。 • **Microsoft OLE DB Provider for SQL Server (OLEDBSQL)** • **SQL Server Native Client (SQLNCLI)**
他のデータベース	データベースに適用可能なプロバイダーを選択してください。 視聴中のデータベースへのOLE DB プロバイダーが使用できない場合 データベースベンダーから必要なドライバーをインストールします（ データベースドライバー概要 575 を参照してください）。代わりにADO.NET、ODBC、またはJDBC 接続を設定します。 オペレーティングシステムが必要なデータベースのためにODBC ドライバーを必要とする場合 **Microsoft OLE DB Provider for ODBC Drivers** を使用することもできます。またはODBC 接続 591 を使用してください。

5. プロバイダーを選択後 [次へ] をクリックしてウィザードを完了します。

ウィザードの次のステップは選択されたプロバイダーにより異なります。SQL Server には、データベースサーバーのホスト名、認証メソッド、データベース名、およびデータベースユーザー名とパスワードを提供または選択する必要があります。サンプルに関してはMicrosoft SQL Server（ADO）への接続 627 セクションを参照ください。Microsoft Access に関しては、データベースファイルへのパスを参照または提供するように問われます。サンプルに関してはMicrosoft Access（ADO）への接続 624 セクションを参照ください。

初期化プロパティの完全なリスト（接続パラメーター）は接続ダイアログボックスの全てのタブで使用可能です。これらのプロパティは選択されたプロバイダーにより異なり、接続を可能にするために明示的に設定される必要がある場合があります。以下のセクションはMicrosoft Access とSQL Server データベースのための基本的な初期化プロパティを構成する方法について説明しています。

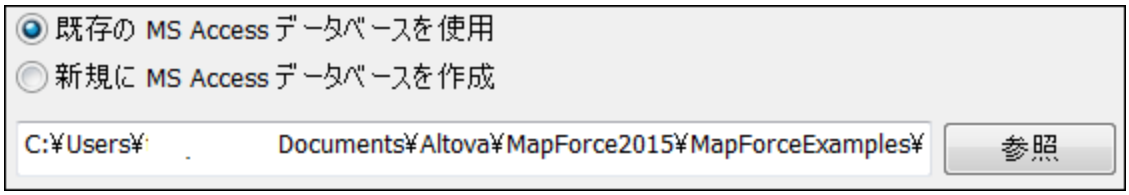
- SQL Server データリンクプロパティの設定 582
- Microsoft Access データリンクプロパティの設定 583

16.1.3.1 既存の Microsoft Access データベースへの接続

このアプローチは、パスワードで保護されていないMicrosoft Access データベースに接続する場合に適しています。データベースがパスワードにより保護されている場合、Microsoft Access（ADO）への接続 624 への接続で説明されている通りデータベースパスワードを設定してください。

既存のMicrosoft Access データベースへの接続方法:

1. データベース接続ウィザードを実行します（ データベース接続ウィザードの開始 574 を参照）。
2. **Microsoft Access (ADO)** を選択して、「次へ」をクリックします。



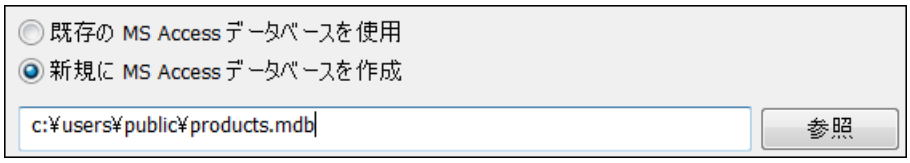
3. [既存のMS Access データベースを使用] を選択します。
4. データベースファイルを参照、または、(相対または絶対)パスを入力します。
5. 「接続」をクリックします。

16.1.3.2 新規の Microsoft Access データベースの作成

既存のデータベースファイルに接続する代替として、Microsoft Access が使用中のマシンにインストールされていない場合でも、新規の Microsoft Access データベースファイル(.accdb, .mdb) を作成して、接続することができます。XMLSpy により作成されるデータベースは空です。Microsoft Access または、DatabaseSpy (http://www.altova.com/ja/databasespy.html)などのツールを使用して、必要とされるデータベース構造を作成し、データを追加します。

新規の Microsoft Access データベースの作成方法:

1. データベース接続ウィザードを実行します (データベース接続ウィザードの開始 574 を参照)。
2. **Microsoft Access (ADO)** を選択して、「次へ」をクリックします。



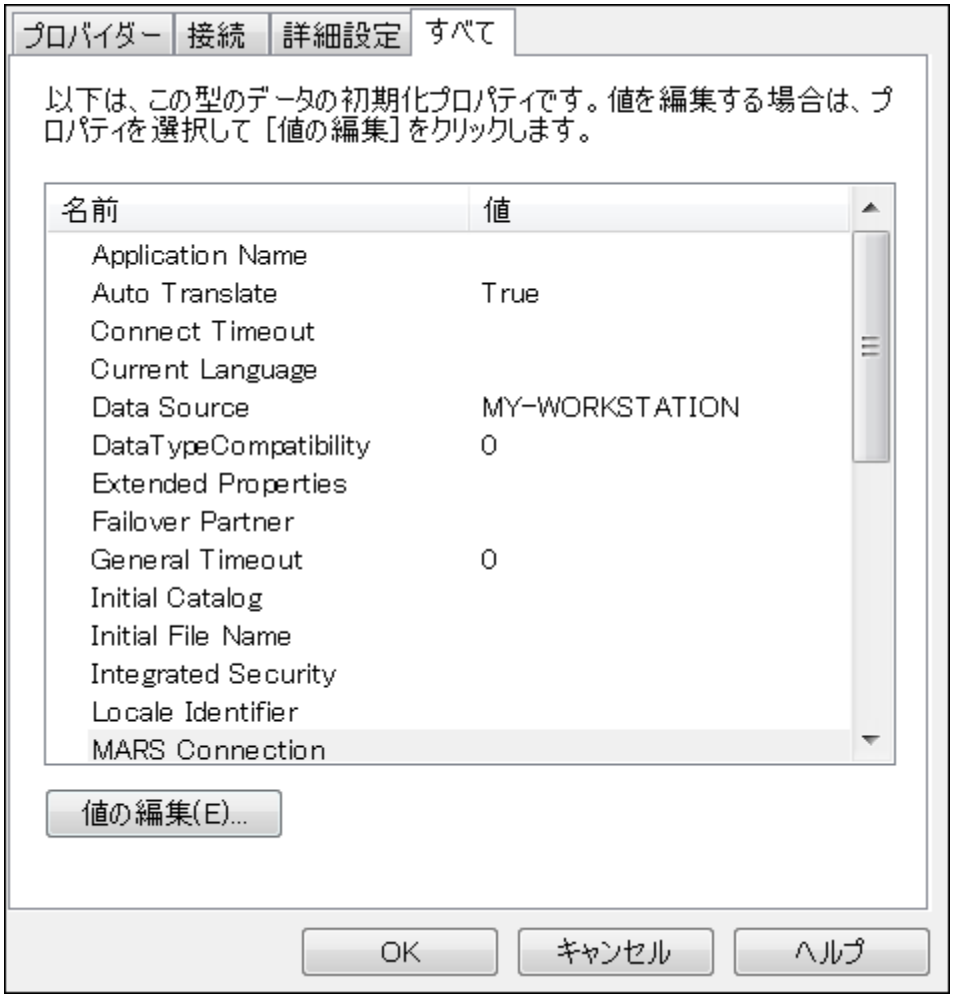
3. [新規 MS Access データベースの作成] を選択して、作成されるデータベースファイルの(相対または絶対)パスを入力します(例えば、**c:\users\public\products.mdb**)。または、[参照] をクリックして、フォルダーを選択して、ファイル名テキストボックスにデータベースファイルの名前(例えば、**products.mdb**)を入力し、[保存] をクリックします。

> **メモ**
> - データベースファイルを作成する場所に対して、書き込みアクセスの許可があることを確認してください。
> - データベースファイル名は **.mdb** or **.accdb** 拡張子を持つ必要があります。

4. 「接続」をクリックします。

16.1.3.3 SQL Server データ リンク プロパティの設定

Microsoft SQL Server データベースに ADO 578 を介して接続する場合、次のデータリンク プロパティダイアログボックスの全ての タブ内の次の接続プロパティを設定する必要があります。

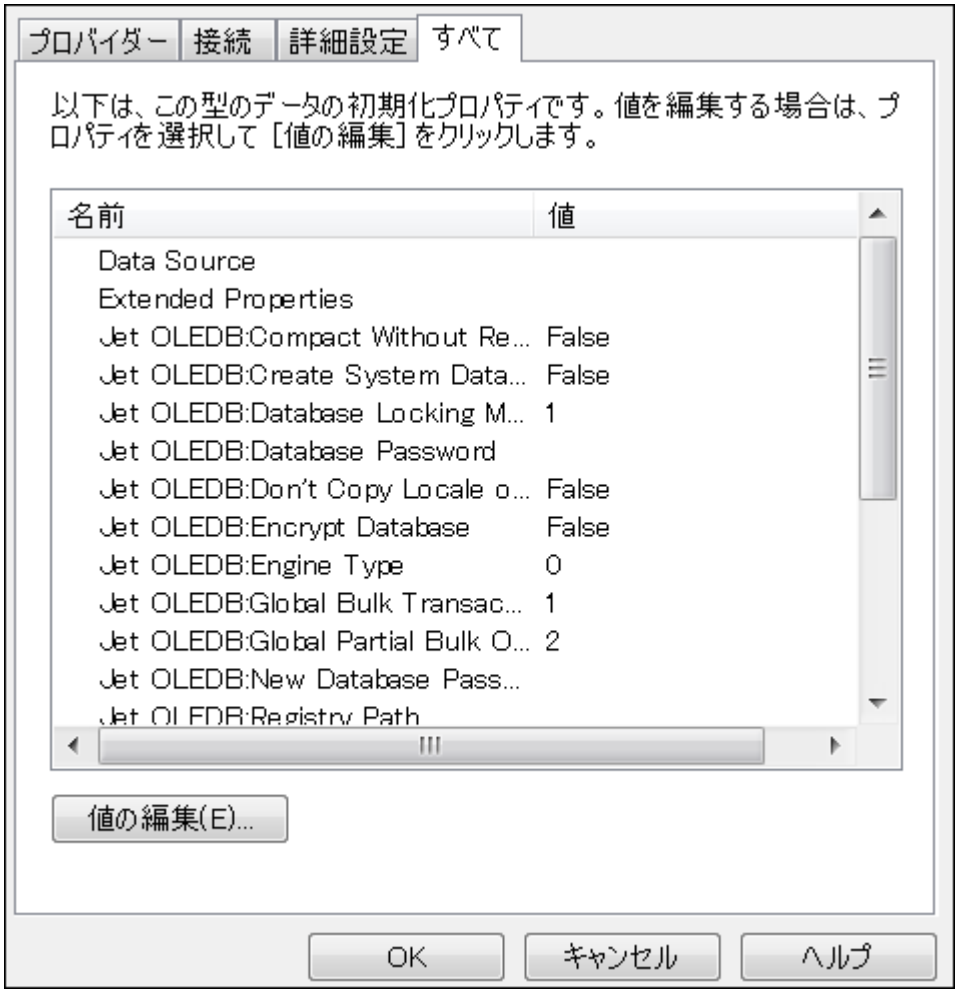


データ リンク プロパティ ダイアログ ボックス

プロパティ	メモ
統合セキュリティ	プロバイダータブで **SQL Server Native Client** がデータプロバイダーに選択された場合は、このプロパティを、空白文字に設定します。
セキュリティ情報を保持	このプロパティを**True** に設定する。

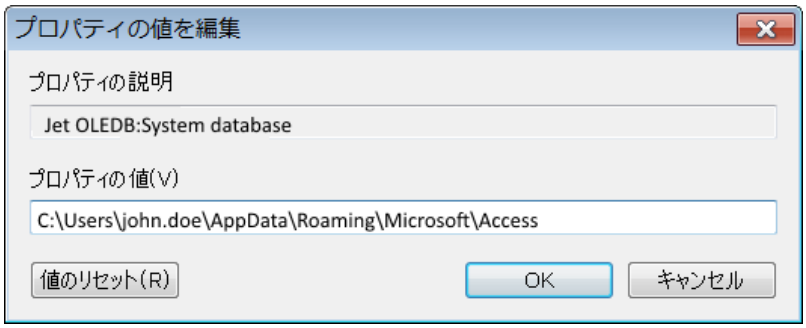
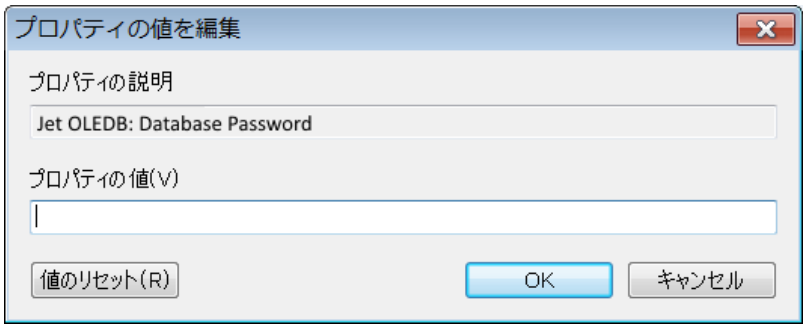
16.1.3.4 Microsoft Access データ リンク プロパティの設定

Microsoft Access データベースにADO[578] を介して接続する場合、次のデータリンク プロパティダイアログボックスの全ての タブ内の次の接続プロパティを設定する必要があります。



データリンクプロパティダイアログボックス

プロパティ	メモ
データソース	このプロパティは、Microsoft Access データベースファイルへのパスを保管します。データベースへの接続問題を回避するために、UNC (Universal Naming Convention 汎用名前付け規則) パスフォーマットの使用が奨励されます。例: `\\anyserver\share$\filepath`
Jet OLEDB: システムデータベース	このプロパティはワークグループ情報ファイルへのパスを保管します。データベースに接続する前に、このプロパティを明示的に設定する必要がある可能性があります。 "ワークグループ情報ファイル" エラーが理由で接続できない場合、ワークグループ情報ファイル (**System.MDW**) をユーザープロファイルに適用できるように設定します (手順は http://support.microsoft.com/kb/305542 を参照してください)、プロパティの値を **System.MDW** ファイルのパスに設定します。

プロパティ	メモ
	
Jet OLEDB: データベース パスワード	データベースがパスワードで保護されている場合は、このプロパティの値をデータベースのパスワードに設定します。 

16.1.4　ADO.NET 接続のセットアップ

ADO.NET は、データベースからのデータを含むデータと作業するためにデザインされたMicrosoft .NET フレームワークライブラリのセットです。XMLSpy から、データベースに接続するには、ADO.NET、Microsoft .NET フレームワーク4、または、以降を使用する必要があります。下に表示されているとおり、 .NET プロバイダーを選択して、接続文字列を提供し、データベースにADO.NET を介して接続します。

.NET データプロバイダーは、コマンドを実行する特定の型のデータソースへの接続を可能にするクラスのコネクションです（例えば、SQL Server またはOracle データベースなど）。すなわち、ADO.NET とは、XMLSpy などのアプリケーションがデータベースプロバイダーを介してデータベースと作動することを意味します。各 データプロバイダーは特定の型のデータソースと作動するように最適化されています。.NET プロバイダーには以下の2 つの種類があります：

1. デフォルトとしてMicrosoft .NET フレームワークを使用して提供されます。
2. 主要なデータベースベンダーを使用して.NET フレームワークの拡張として提供されます。このようなADO.NET プロバイダーは、個別にインストールされている必要があり、通常それぞれのデータベースベンダーのウェブサイトからダウンロードすることができます。

メモ　特定のADO.NET プロバイダーは、サポートされていない、または、サポートが制約されています。ADO.NET サポートに関するメモ 590 を参照してください。

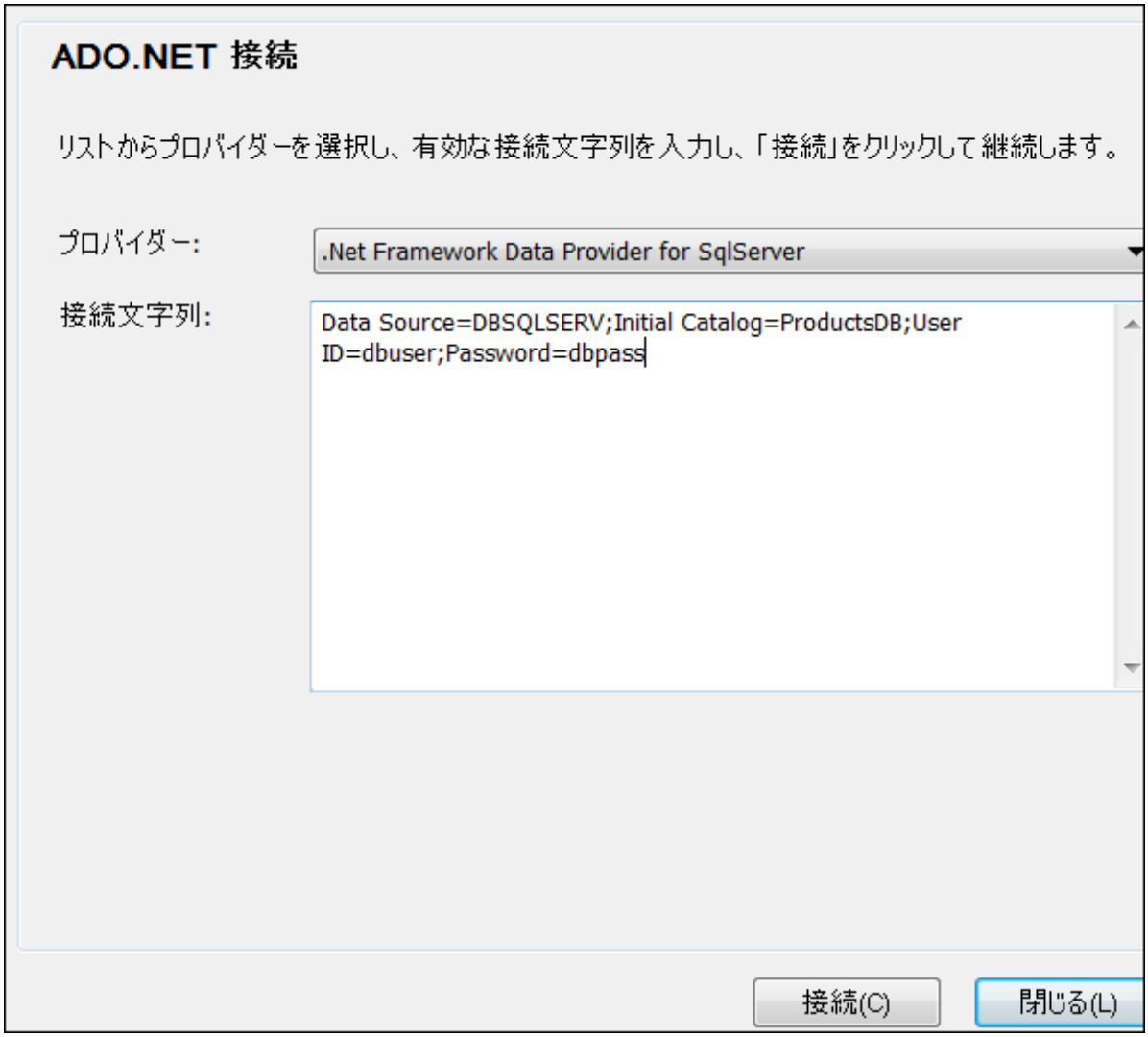
ADO.NET 接続の作成:

1. データベース接続ウィザードを開始します 574 。
2. 「ADO.NET 接続」をクリックします。
3. .NET データプロバイダーをリストから選択します。

.NET フレームワークと共に使用することのできるプロバイダーのリストはデフォルトでプロバイダーのリストに表示されます。ベンダーに特有の.NET データプロバイダーは、使用中のシステムに既にインストールされている場合のみ使用することができます。.NET データプロバイダーは、データベースベンダーにより提供される、.msi、または、.exe ファイルを実行して、GAC (Global Assembly Cache) 内にインストールされる必要があります。

4. データベース接続文字列を入力します。接続文字列は、データベース接続情報をセミコロンで区切られた接続パラメーターのキー値ペアとして定義します。例えば、**`Data Source=DBSQLSERV;Initial Catalog=ProductsDB;User ID=dbuser;password=dbpass`** などの文字列は、サーバー `DBSQLSERV` 上で、ユーザー名 `dbuser` とパスワード `dbpass` を使用して、SQL Server データベース `ProductsDB` に接続します。接続文字列を、キー値ペアを直接「接続文字列」ダイアログボックスに入力することにより作成することができます。他の接続オプションは Visual Studio で作成することができます (次を参照: Visual Studio 内で接続文字列を作成する 587)。

接続文字列の構文は、プロバイダーリストより選択されたプロバイダーにより異なります。例えば、次を参照してください: サンプルADO.NET 接続文字列 589。



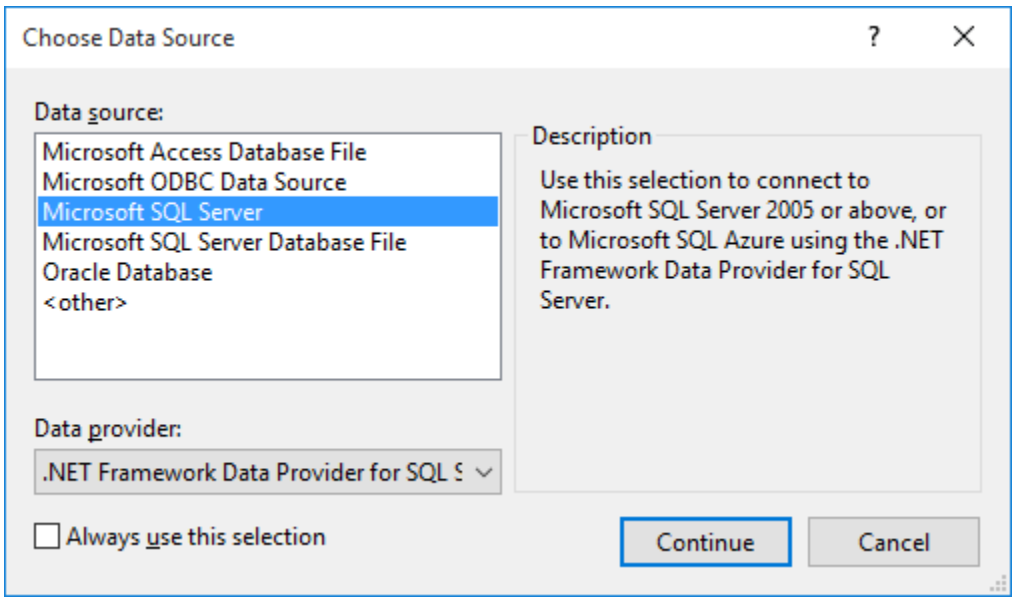
5. 「接続」をクリックします。

16.1.4.1 Visual Studio 内で接続文字列を作成する

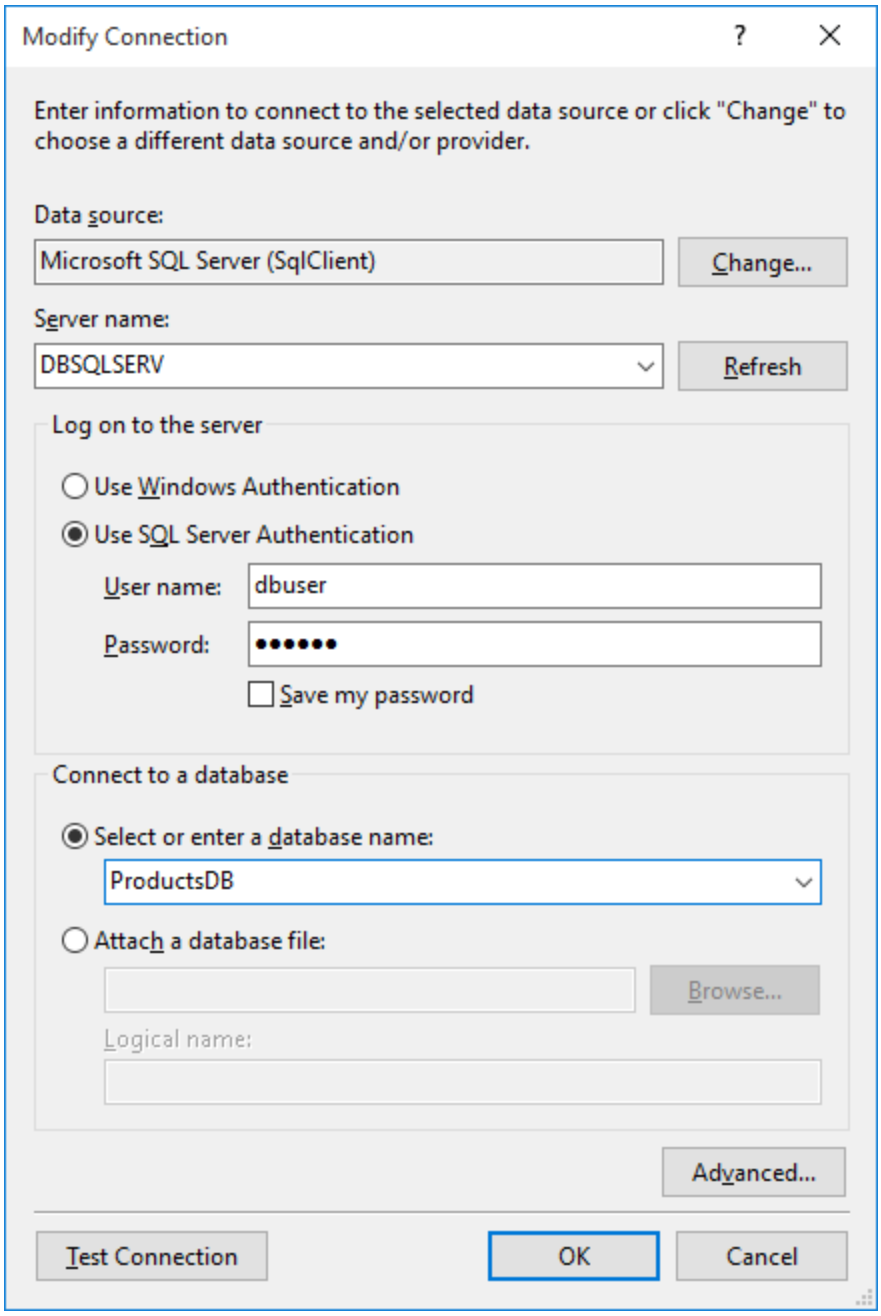
ADO.NET を使用してデータソースに接続するには、有効な接続文字列が必要です。次の手順は、Visual Studio から接続文字列を作成する方法を説明しています。

Visual Studio 内で接続文字列を作成する:

1. 「ツール」メニューから、「データベースへの接続」をクリックします。
2. リストからデータソースを選択します(この例の場合、Microsoft SQL Server)。データプロバイダーは、選択に従い自動的に記入されます。



3. 「接続」をクリックします。



4. データベースへのサーバーホスト名、ユーザー名、およびパスワードを入力します。この例の場合、サーバー DBSQLSERV 上のデータベース ProductsDB にSQL Server 認証を使用して接続します。

5. 「OK」をクリックします。

データベースへの接続に成功した場合、サーバーエクスプローラーウィンドウに表示されます。サーバーエクスプローラーウィンドウを、メニューコマンド「**表示 | サーバーエクスプローラー**」を使用して表示することができます。データベース接続文字列を取得するには、サーバーエクスプローラーウィンドウ内の接続を右クリックし、「**プロパティ**」を選択します。接続文字列が、Visual Studio プロパティウィンドウに表示されます。XMLSpy の「接続文字列」ボックスに文字列を貼り付ける前に、アスタリスク（＊）文字を実際のパスワードと置き換える必要があることに注意してください。

16.1.4.2 サンプル ADO.NET 接続文字列

ADO.NET 接続をセットアップするには、データベース接続ダイアログボックスからADO.NET プロバイダーを選択し、接続文字列を入力します。（次も参照してください ADO.NET 接続のセットアップ 585）。異なるデータベースのサンプルADO.NET 接続文字列は.NET プロバイダーの下にリストされています。

Teradata のための.NET Data

このプロバイダーは、次のTeradata Web サイトからダウンロードすることができます (http://downloads.teradata.com/download/connectivity/net-data-provider-for-teradata)。サンプル接続は以下のとおりです:

```
Data Source=ServerAddress;User Id=user;Password=password;
```

IBM i のための.NET Framework データプロバイダー

このプロバイダーは、*IBM i Access Client Solutions - Windows Application Package* の一部としてインストールされます。サンプル接続文字列は以下のとおりです:

```
DataSource=ServerAddress;UserID=user;Password=password;DataCompression=True;
```

詳細に関しては、インストールパッケージ内に含まれる「.NET プロバイダー技術レファレンス ヘルプファイル」を参照してください。

MySQL のための.NET Framework データプロバイダー

このプロバイダーをMySQL ウェブサイトからダウンロードすることができます（https://dev.mysql.com/downloads/connector/net/）。サンプル接続文字列は以下のとおりです:

```
Server=127.0.0.1;Uid=root;Pwd=12345;Database=test;
```

次も参照してください: https://dev.mysql.com/doc/connector-net/en/connector-net-programming-connecting-connection-string.html

SQL Server のための.NET Framework データプロバイダー

サンプル接続文字列は以下のとおりです:

```
Data Source=DBSQLSERV;Initial Catalog=ProductsDB;User ID=dbuser;Password=dbpass
```

次も参照してください: https://msdn.microsoft.com/en-us/library/ms254500(v=vs.110).aspx

.NET Framework 4.0 のためのIBM DB2 データプロバイダー10.1.2

```
Database=PRODUCTS;UID=user;Password=password;Server=localhost:50000;
```

メモ このプロバイダーは、通常 IBM DB2 Data Server Client パッケージと共にインストールされます。IBM DB2 Data Server Client パッケージをインストール後 ADO.NET プロバイダーのリスト内でプロバイダーが不足している場合、次の技術メモを参照してください: http://www-01.ibm.com/support/docview.wss?uid=swg21429586。

次も参照してください:
http://www.ibm.com/support/knowledgecenter/en/SSEPGG_10.1.0/com.ibm.swg.im.dbclient.adonet.ref.doc/doc/DB2ConnectionClassConnectionStringProperty.html

.NET (ODP.NET) のための Oracle データプロバイダー

ODP.NET プロバイダーを含むインストールパッケージを Oracle ウェブサイトからダウンロードすることができます (次を参照してください: http://www.oracle.com/technetwork/topics/dotnet/downloads/index.html). サンプル接続文字列は以下のとおりです:

```
Data Source=DSORCL;User Id=user;Password=password;
```

Oracle への接続(ODBC) 638 内で説明されているように **tnsnames.ora** ファイル内で定義された Oracle サービス名をさす DSORCL がデータソース名の場合。

tnsnames.ora ファイル内のサービス名を構成せず接続する場合は、以下のような文字列を使用してください:

```
Data Source=(DESCRIPTION=(ADDRESS_LIST=(ADDRESS=(PROTOCOL=TCP)(HOST=host)(PORT=port)))
(CONNECT_DATA=(SERVER=DEDICATED)(SERVICE_NAME=MyOracleSID)));User
Id=user;Password=password;
```

次も参照してください: https://docs.oracle.com/cd/B28359_01/win.111/b28375/featConnecting.htm

16.1.4.3 ADO.NET サポートに関するメモ

下のテーブルはXMLSpy 内で現在サポートされていない、または、制限のあるサポートのある既知のADO.NET データベースドライバーがリストされています。

データベース	ドライバー	サポートノート
全てのデータベース	ODBC のための Net Framework **Data Provider**	制限つきサポート。Microsoft Access 接続の際に既知の問題があります。ODBC への直接の接続を代わりに使用することが奨励されます。
	OleDb のための .Net Framework **Data Provider**	制限つきサポート。Microsoft Access 接続の際に既知の問題があります。ADO への直接の接続を代わりに使用することが奨励されます。
Firebird	**Firebird ADO.NET Data Provider**	制限つきサポート。ODBC または JDBC を代わりに使

データベース	ドライバー	サポートノート
		用することが奨励されます。
Informix	.NET Framework 4.0 のための IBM **Informix Data Provider**	サポートされていません。**DB2 Data Server Provider** を代わりに使用してください。
IBM DB2 for i (iSeries)	i5/OS のための .Net Framework **Data Provider**	サポートされていません。*IBM i Access Client Solutions - Windows Application Package* の一部として提供されている **.Net Framework Data Provider for IBM i** を代わりに使用してください。
Oracle	Oracle のための .Net Framework **Data Provider**	制限つきサポート。.NET Framework と共にこのドライバは提供されていますが、Microsoft により使用を奨励されていません。
PostgreSQL	-	このベンダーのための ADO.NET ドライバはサポートされていません。ネイティブな接続を代わりに使用してください。
Sybase	-	このベンダーのための ADO.NET ドライバはサポートされていません。

16.1.5 ODBC 接続の設定

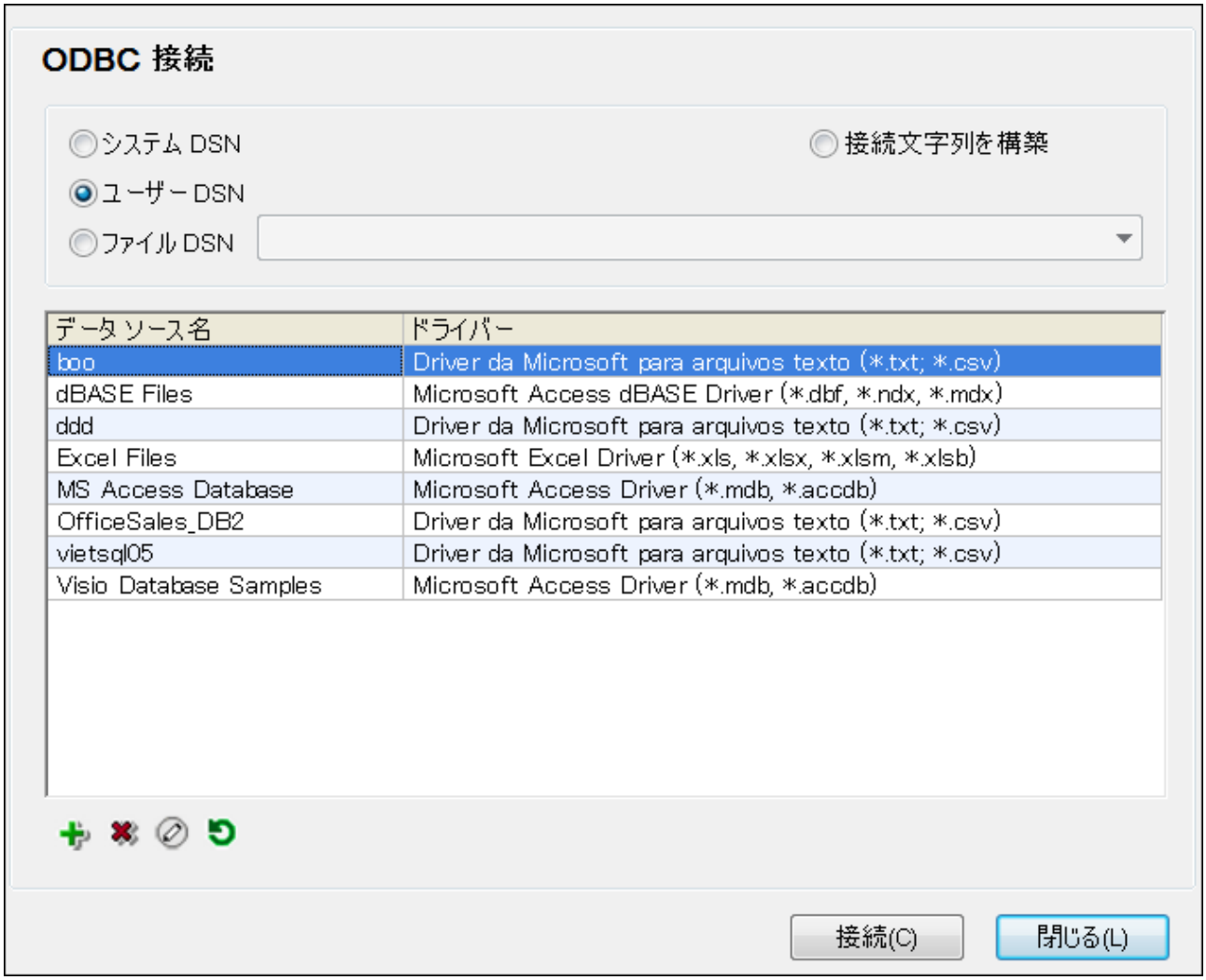
ODBC (Open Database Connectivity) は、一般に使用されているデータアクセス技術で、XMLSpy からデータベースに接続することができます。データベースにアクセスするための主要な手段として、または ネイティブな OLE DB- や JDBC-による接続の代替として使用することができます。

ODBC を使用してデータベースに接続するには、最初に ODBC データソース名(DSN) を作成する必要があります。このステップは、他のユーザーまたはオペレーティングシステムにより DSN が既に作成されている場合必要ありません。DSN は、統一された、XMLSpy を含むオペレーティングシステム上の ODBC に対応したクライアントアプリのデータベース接続方法を表します。DSN には以下の種類があります:

- ファイル DSN
- ユーザー DSN
- ファイル DSN

オペレーティングシステムで特権を持つすべてのユーザーはシステム データ ソースにアクセスすることができます。作成したユーザーのみがユーザーデータ ソースにアクセスすることができます。*ファイル DSN* を作成した場合、データソースは .dsn 拡張子を持つファイルとして作成され、データソースにより使用されているドライバーがインストールされていることが前提で、他のユーザーと共有することができます。

ODBC 接続ダイアログボックス内の [**ODBC 接続**] をクリックすると、使用中のコンピューターで既に使用することのできる DSN はデータベース接続ダイアログボックスに表示されています。



ODBC 接続ダイアログボックス

データベースにより必要とされるDSN が無い場合、XMLSpy データベース接続ウィザードが作成の手助けをします。しかし、W indows オペレーティングシステムに直接作成することもできます。いずれにせよ、続行する前に、使用中のデータベースにODBC ドライバーがオペレーティングシステムのODBC ドライバーリストに適用できることを確認してください。（ 使用可能な ODBC ドライバーの表示 593 を参照）。

新規のDSNを使用しての接続:

1. データベース接続ウィザードを開始します 574 。
2. データベース接続ダイアログボックスの[ODBC 接続] をクリックします。
3. システムDSN を作成するには、オペレーティングシステムの管理者権限が必要です。

> また、XMLSpy が管理者として作動している必要があります。

4. 「追加」 をクリックします。

5. ドライバーを選択して、(作成するDSN の種類に応じて) [**ユーザーDSN**] または[**システムDSN**] をクリックします。使用中のデータベースに適用することのできるドライバーがリストされていない場合、データベースベンダーからダウンロードして、インストールしてください。(次を参照: データベースドライバー概要)。
6. ポップアップするダイアログボックスに特定の接続情報を入力してセットアップを完了します。

接続を成功させるには、データベースサーバーのホスト名(またはIP アドレス)、とデータベースのユーザー名およびパスワードを提供する必要があります。他の接続パラメーターがある可能性がありますが、これらのパラメーターはデータベースプロバイダーにより異なります。各接続方法に固有のパラメーターについての詳細は、ドライバープロバイダーのドキュメントを参照してください。一度作成されると、DSN はデータソース名のリストで使用することができます。これによりデータベース接続の詳細をデータベースの接続を希望する際いつでも使用することができます。ユーザーDSN はユーザーDSN のリストに追加されますが、システムDSN はシステムDSN のリストに追加されます。

既存のDSNを使用しての接続:

1. データベース接続ウィザードを開始します[574]。
2. [**ODBC 接続**] をクリックします。
3. 既存のデータソース(ユーザーDSN、システムDSN、ファイルDSN) から種類を選択します
4. 既存 DSN レコードをクリックして、「**接続**」をクリックします。

既存の.dsn ファイルをベースにして接続文字列を構築する:

1. データベース接続ウィザードを開始します[574]。
2. [**ODBC 接続**] をクリックします。
3. [**接続文字列の構築**] を選択して [構築] をクリックします。
4. ファイルDSN を使用して、接続文字列を構築する場合、[**ファイルデータソース**] タブをクリックします。それ以外の場合は、[**マシンデータソース**] タブをクリックします。(システムDSN とユーザーDSN は"マシン" データソースとして知られています。)
5. 必要とされる.dsn ファイルを選択して、「OK」をクリックします。

用意された接続文字列を使用して接続する:

1. データベース接続ウィザードを開始します[574]。
2. [**ODBC 接続**] をクリックします。
3. [**接続文字列の構築**] を選択しています。
4. 接続文字列を与えられたボックスに貼り付け、「**接続**」をクリックします。

16.1.5.1 使用可能な ODBC ドライバーの表示

ODBC データソース管理で使用中のオペレーティングシステムで使用可能な ODBC ドライバーを確認することができます。Windows コントロールパネルの**管理ツール**から ODBC データソース管理 (**Odbcad32.exe**) にアクセスすることができます。64-ビット版オペレーティングシステムには、2 種類のバージョンの実行可能ファイルがあります:

- **C:\Windows\SysWoW64** ディレクトリにある、**Odbcad32.exe** ファイルの32-ビット版バージョン(**C:** が使用中のシステムドライブとして前提されています)。
- **C:\Windows\System32** ディレクトリにある、**Odbcad32.exe** ファイルの64-ビット版バージョン

インストールされている32-ビット版データベースドライバーは、ODBC データソース管理の32-ビット版バージョンで、64-ビット版バージョン内の64-ビット版ドライバーで確認することができます。ですから、対応するバージョンのODBC データソース管理でデータベースドライバーを確認してください。

ODBC データソース管理者

ターゲットデータベースのドライバーがリストに存在しない場合、または、代替ドライバーを追加する場合、データベースのベンダーからダウンロードしてください（データベースドライバー概要 575 を参照）。ODBC ドライバーがシステムで使用可能になると、ODBC 接続を作成することができます（ODBC 接続のセットアップ 591 を参照）。

16.1.6 JDBC 接続の設定

JDBC（Java Database Connectivity）は、Oracle の Java ソフトウェアプラットフォームの一部であるデータベースアクセスインターフェイスです。JDBC 接続は ODBC 接続に比べ、通常より少ないリソースを集中的に使用する接続ですが、ODBC で使用することのできない機能を使用することができます。ODBC コネクタにより使用できないデータベース機能を使用する場合 JDBC 接続を使用することが奨励されています。

前提条件

- JRE（Java Runtime Environment）または Java Development Kit（JDK）がインストールされている必要があります。これは Oracle JDK または Oracle OpenJDK などのオープンソースビルドであることができます。XMLSpy は次のロケーションから Java Virtual Machine（JVM）へのパスを次の順序で決定します: a）アプリケーション オプション内で設定したカスタム JVM パス b）Windows レジストリ内で見つかった JVM パス。c）`JAVA_HOME` 環境変数。
- XMLSpy（32-ビット、64-ビット）のプラットフォームが JRE/JDK のプラットフォームに一致することを確認してください。
- データベースベンダーからの JDBC ドライバーがインストールされている必要があります。これは使用可能でありデータベースによりサポートされている場合、データベースクライアントインストールの一部として、または、JDBC ライブラリ（.jar ファイル）として個別にダウンロードされている JDBC ドライバーの可能性があります。データベース接続のサンプル 603 も参照してください。
- Oracle データベースに接続する場合は、一部の Oracle ドライバーは特定の JRE バージョンに固有のため、追加コンポーネントと構成が必要な点に注意してください。Oracle 製品のドキュメント（例えば、「JDBC 開発者ガイドおよびリファレンス」）には、各 JDBC ドライバーの構成の手続きの詳細が説明されています。

- CLASSPATH 環境変数は、(.jar ファイルの1つまたは複数の) 使用中のWindows オペレーティングシステムのJDBC ドライバーのパスを含む必要があります。データベースクライアントをインストールする際に、インストーラーはこの変数を自動的に構成する可能性があります (CLASSPATH の構成 596 を参照)。

Windows 資格情報を使用してJDBC を介してSQL Server に接続する方法

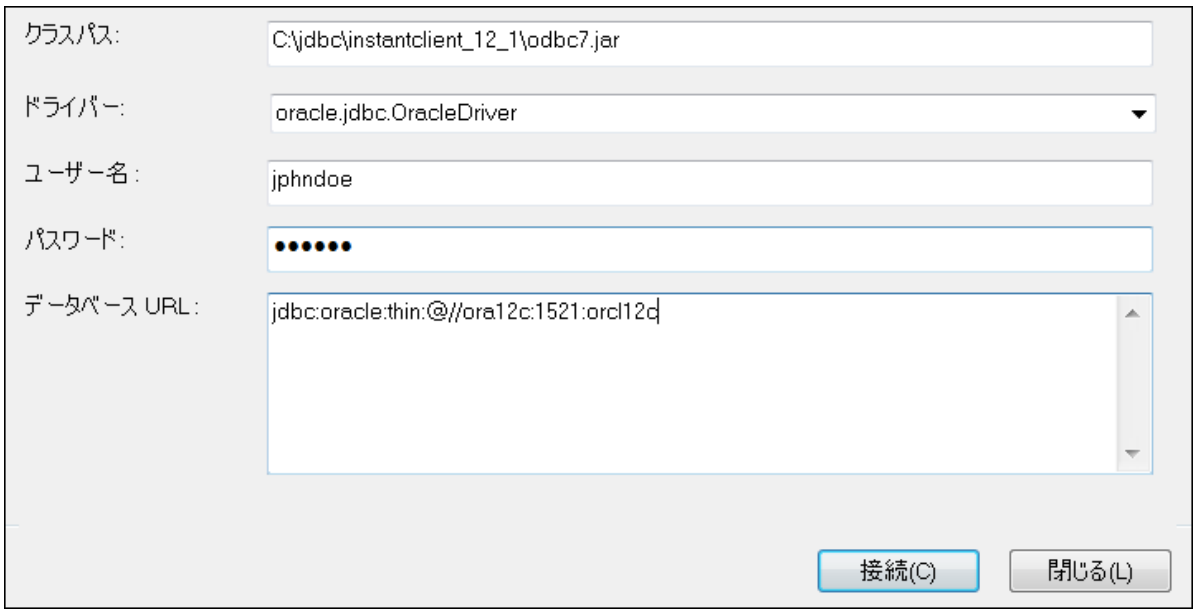
Windows 資格情報 (統合されたセキュリティ) を使用してJDBC を介してSQL Serverに接続する場合、以下に注意してください:

- JDBC ドライバーパッケージに含まれている **sqljdbc_auth.dll** ファイルはシステムPATH 環境変数上のディレクトリにコピーされる必要があります。このようなファイルは2個存在します。1つはx86 プラットフォームのために、もう1つはx64 プラットフォームのために存在します。JDK プラットフォームに対応するPATH が追加されていることを確認してください。
- JDBC 接続文字列はプロパティ `integratedSecurity=true` を含む必要があります。このプロパティを異なる場所から追加することができます:
  - データベース接続ウィザードから追加することができます。以下を参照してください。
  - データベースコンポーネント設定から追加することができます。
  - 適応可能な場合、生成済みのJava コード内のデータベース接続文字列の編集をすることにより追加することができます。

詳細に関しては *SQL Server のための Microsoft JDBC ドライバードキュメント*を参照してください。
https://docs.microsoft.com/en-us/sql/connect/jdbc/building-the-connection-url。

JDBC 接続のセットアップ方法

1. データベース接続ウィザードを開始します。 574
2. 「**JDBC 接続**」をクリックします。
3. 任意で、セミコロンで区切られた.jar ファイルパスのリストを「クラスパス」テキストボックスに入力することもできます。ここで入力される.jar ライブラリは、CLASSPATH 環境変数内で既に定義されているものに加え、環境にロードされます。「クラスパス」テキストボックスの編集が完了すると、ソース.jar ライブラリ内で検索されるJDBC ドライバーは、「ドライバー」リストに自動的に追加されます (次のステップを参照してください)。



4. 「ドライバー」の横でJDBC ドライバーをリストから選択、または、Java クラス名を入力します。CLASSPATH 環境変数により構成されたJDBC ドライバーおよび、「クラスパス」テキストボックス内で検出されたドライバーが含まれます (次を参照: CLASSPATH の構成 596)。

CLASSPATH 変数内で定義されているJDBC ドライバーのパスを使用することができます。また、データベース接続ダイアログボックスに直接入力された.jar パスはJava Virtual Machine（JVM）によりサポートされています。JVM は接続を構築するためのドライバーを決定します。JVM にロードされるJava クラスを確認し、JDBC ドライバーの競合、および、データベースに接続する際に発生する予期されないエラーを回避するようにしてください。

5. 対応するボックスに、ユーザー名とパスワードを入力します。
6. データベースURL テキスト・ボックス内に、JDBC 接続 URL（文字列）をデータベースの型のとおり入力します。次のテーブルは一般のデータベースの型のためのJDBC 接続 URL（文字列）の構文について説明しています。

データベース	JDBC 接続 URL
Firebird	jdbc:firebirdsql://**<host>**[:**<port>**]/**<database path or alias>**
IBM DB2	jdbc:db2://**hostName**:**port**/**databaseName**
IBM DB2 for i	jdbc:as400://**[host]**
IBM Informix	jdbc:informix-sqli://**hostName**:**port**/**databaseName**:INFORMIXSERVER=**myserver**
MariaDB	jdbc:mariadb://**hostName**:**port**/**databaseName**
Microsoft SQL Server	jdbc:sqlserver://**hostName**:**port**;**databaseName**=name
MySQL	jdbc:mysql://**hostName**:**port**/**databaseName**
Oracle	jdbc:oracle:thin:@**hostName**:**port**:**SID** jdbc:oracle:thin:@//**hostName**:**port**/**service**
Oracle XML DB	jdbc:oracle:oci:@//**hostName**:**port**:**service**
PostgreSQL	jdbc:postgresql://**hostName**:**port**/**databaseName**
Progress OpenEdge	jdbc:datadirect:openedge://**host**:**port**;databaseName=**db_name**
Sybase	jdbc:sybase:Tds:**hostName**:**port**/**databaseName**
Teradata	jdbc:teradata://**databaseServerName**

メモ 上記のフォーマットに対する構文のバリエーションも使用することができます（例えば、データベースURL はポートを除外、またはデータベースのユーザー名とパスワードを含む可能性があります）。詳細はデータベースのベンダーのドキュメントを参照してください。

7. 「接続」をクリックします。

16.1.6.1 CLASSPATH の構成

CLASSPATH 環境変数はクラスと、オペレーティングシステム上の他のソースファイルを検索するためJava Runtime Environment（JRE）またはJava Development Kit（JDK）により使用されます。JDBC を介してデータベースに接続すると、この変数はオペレー

ティングシステム上のJDBC ドライバーへのパス、または一部の場合、使用中のデータベースの種類に固有の追加ライブラリファイルへのパスを含むように構成されなければなりません。

下のテーブルは、CLASSPATH 変数に通常含まれる必要のあるサンプルファイルパスをリストしています。重要なことは、システム上の情報をシステム上のJDBC ドライバーのロケーション、JDBC ドライバー名、およびオペレーティングシステム上に存在するJRE/JDK バージョンをベースにしたこの情報を調整する必要があります。接続問題を避けるために、使用中のオペレーティングシステムにインストールされているJDBC ドライバに適用されるインストールの指示、インストール前とインストール後の構成ステップを確認してください。

データベース	サンプルCLASSPATH エントリ
Firebird	C:\Program Files\Firebird\Jaybird-2.2.8-JDK_1.8\jaybird-full-2.2.8.jar
IBM DB2	C:\Program Files (x86)\IBM\SQLLIB\java\db2jcc.jar;C:\Program Files (x86)\IBM\SQLLIB\java\db2jcc_license_cu.jar;
IBM DB2 for i	C:\jt400\jt400.jar;
IBM Informix	C:\Informix_JDBC_Driver\lib\ifxjdbc.jar;
Microsoft SQL Server	C:\Program Files\Microsoft JDBC Driver 4.0 for SQL Server\sqljdbc_4.0\enu\sqljdbc.jar
MariaDB	**<installation directory>**\mariadb-java-client-2.2.0.jar
MySQL	**<installation directory>**\mysql-connector-java-*version*-bin.jar;
Oracle	**ORACLE_HOME**\jdbc\lib\ojdbc6.jar;
Oracle (with XML DB)	**ORACLE_HOME**\jdbc\lib\ojdbc6.jar;**ORACLE_HOME**\LIB\xmlparserv2.jar;**ORACLE_HOME**\RDBMS\jlib\xdb.jar;
PostgreSQL	**<installation directory>**\postgresql.jar
Progress OpenEdge	%DLC%\java\openedge.jar;%DLC%\java\pool.jar; メモProgress OpenEdge SDK がマシンにインストールされていると想定します。%DLC% は OpenEdge がインストールされているディレクトリです。
Sybase	C:\sybase\jConnect-7_0\classes\jconn4.jar
Teradata	**<installation directory>**\tdgssconfig.jar;**<installation directory>**\terajdbc4.jar

- CLASSPATH 変数の変更は、使用中のコンピュータのJava アプリケーションの振る舞いに影響する可能性があります。続行する前に生じえる影響を把握するために、Java ドキュメントを参照してください。
- 環境変数は、ユーザーまたはシステムであることができます。システム環境変数を変更するには、オペレーティングシステムに対する管理者権限が必要です。
- 環境変数を変更後、設定の変更を有効にするには、プログラムを再起動してください。または、オペレーティングシステムをログオフまたは再起動してください。

Windows 7 でCLASSPATH を構成する:

1. [スタート] メニューを開始して、[コンピューター] を右クリックします。

2. [プロパティ] をクリックします。
3. [システムの詳細設定] をクリックします。
4. [詳細設定] タブから [環境変数] をクリックします。
5. CLASSPATH 変数を、ユーザーまたはシステム環境変数からロケートして、編集をクリックします。CLASSPATH 変数が存在しない場合は、[新規] をクリックして作成します。
6. オペレーティングシステム内で JDBC ドライバーがある場所にパスを含むように変数の値を編集します。JDBC ドライバーパスを他の既に CLASSPATH 変数内にあるパスから区別するためにセミコロン (;) で区切ります。

Windows 10 で CLASSPATH を構成する:

1. Windows キーを押して、環境編集を入力します。
2. [システム環境変数の編集] の提案をクリックします。
3. [環境変数] をクリックします。
4. CLASSPATH 変数を、ユーザーまたはシステム環境変数からロケートして、編集をクリックします。CLASSPATH 変数が存在しない場合は、[新規] をクリックして作成します。
5. オペレーティングシステム内で JDBC ドライバーがある場所にパスを含むように変数の値を編集します。JDBC ドライバーパスを他の既に CLASSPATH 変数内にあるパスから区別するためにセミコロン (;) で区切ります。

16.1.7 PostgreSQL 接続のセットアップ

PostgreSQL データベースへの接続は、ネイティブな接続として、または、ODBC、JDBC、および他のドライバーを使用して確立することができます。ネイティブな接続の利点は、使用中のシステムにインストールする場合、ドライバーを必要としないことです。

ネイティブでは内ドライバーを使用しての接続を作成する場合は、以下のトピックを参照してください:

- JDBC 接続の設定 594
- PostgreSQL (ODBC) への接続 644

それ以外の場合、PostgreSQL へのネイティブの接続をセットアップする場合、以下の手順に従ってください。継続するには、以下の前提条件が必要です: ホスト名、ポート、データベース名、ユーザー名、および パスワード。

ネイティブ PostgreSQL 接続のセットアップ:

1. データベース接続ウィザードを開始します 574.
2. 「PostgreSQL 接続」をクリックします。
3. 対応するボックスに、ホスト (PostgreSQL が同じマシンで作動している場合、*ローカルホスト*)、ポート (通常、5432 ですが、これは任意です)、データベース名、ユーザー名とパスワードを入力します。

4. 「**接続**」をクリックします。

PostgreSQL データベースサーバーが異なるマシンに存在する場合、次の点に注意してください：

- PostgreSQL データベースサーバーは、クライアントからの接続を受け入れるように構成されなければなりません。具体的には、**pg_hba.conf** ファイルがローカルではない接続を許可するように構成される必要があります。第2に、**postgresql.conf** ファイルは、特定された IP アドレスとポートをリッスンするように構成される必要があります。詳細に関しては、PostgreSQL ドキュメントを参照してください（https://www.postgresql.org/docs/9.5/static/client-authentication-problems.html）。
- サーバーマシンがファイアウォールを越えてデザインされたポート（通常、5432）上で接続を受け入れるように構成されている必要があります。例えば、「**コントロールパネル** > **Windows ファイアウォール** > **詳細設定** > **受信の規則**」で設定すると、Windows 上で作動するデータベースサーバーでは、ファイアーウォールを越えてポート 5432 に接続を許可するルールを作成する必要がない場合があります。

16.1.8 SQLite 接続の設定

SQLite (http://www.sqlite.org) は、ポータブル性とインストールの簡便性が重要な場合理想的な構成のファイルベースの、内蔵のデータ型です。Since SQLite データベースは、XMLSpy によりネイティブにサポートされるため、接続のためにドライバーをインストールする必要はありません。

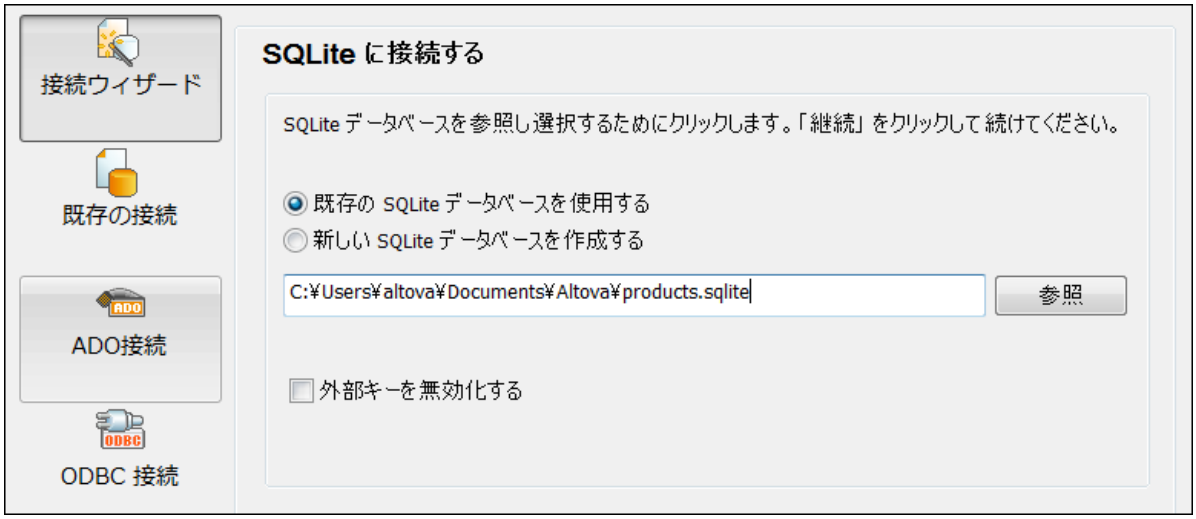
16.1.8.1 既存の SQLite データベースへの接続

接続ウィザードまたは **次を使用して開く...** コマンドを使用して直接ウィンドウエクスプローラーから既存の SQLite データベースに接続することができます。

接続ウィザードの使用

既存のSQLite データベースに接続する方法:

1. データベース接続ウィザードを実行します（ データベース接続ウィザードの開始 574 を参照）。
2. **SQLite** を選択して、「次へ」をクリックします。



3. [**既存のSQLite データベースを使用**] を選択して、SQLite データベースファイルを参照、またはデータベースへの(相対または絶対)パスを入力します。「**接続**」ボタンは、データベースファイルへのパスを入力すると有効化されます。
4. 「**外部キーの無効化**」チェックボックスを任意で選択します。部キーの制約 601 も参照してください。
5. 「**接続**」をクリックします。

ウィンドウエクスプローラーから開く

SQLite データベースをウィンドウエクスプローラーから以下のように直接開くことができます:

1. ウィンドウエクスプローラー内の既存のデータベースファイルを右クリックし、「**開く**」をコンテキストメニューから選択します。
2. **DatabaseSpy** を提案されたリストから選択します。この選択肢が使用できない場合 **他のアプリを選択** を選択し、DatabaseSpy 実行可能ファイルをインストールディレクトリから参照します。
3. データベース接続ウィザードが表示されると [**閉じる**] をクリックして無視します。

後者が既に開かれている場合データベースファイルを DatabaseSpy にドラッグアンドドロップすることができます。

メモ

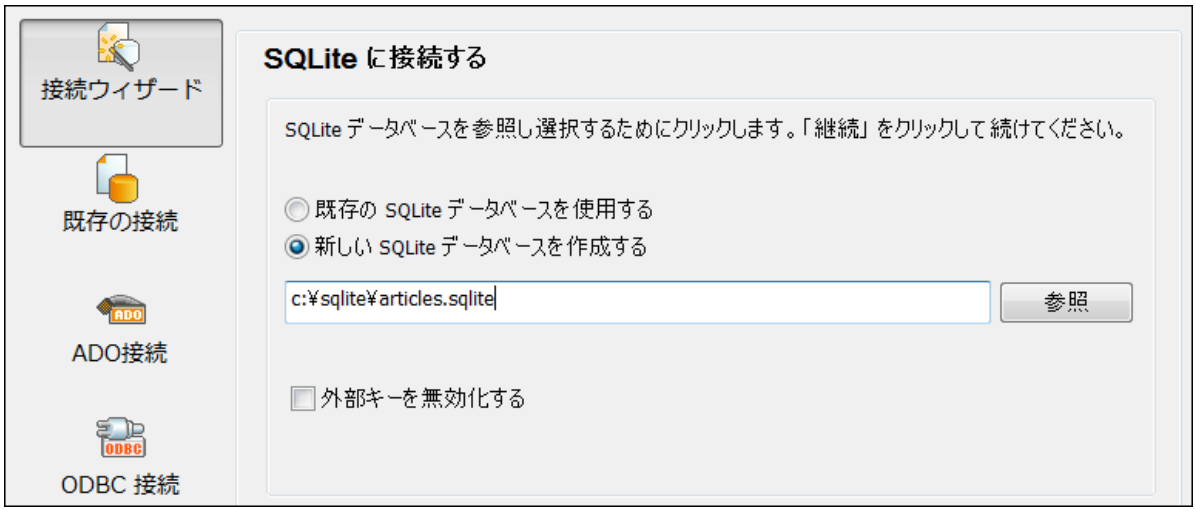
- アプリケーションが起動される都度データベース接続ウィザードが表示されるように [**ツール | オプション**] から DatabaseSpy をオプションで構成することができます。これを行うには設定の **一般** グループのチェックボックス **データベース接続ダイアログの作成を表示** をクリアします。
- DatabaseSpy が開かれており同じデータベースファイルへのデータソースが既存の場合再利用されます。すなわち、同じデータベースへの複製データソースは作成されません。**開く** を使用して、またはドラッグアンドドロップを使用してデータベースファイルを開く際に適用することができます。

16.1.8.2 新規の SQLite データベースの作成

既存のデータベースファイルに接続する代替として、新規のSQLite データベースファイルを作成して、接続することができます。XMLSpy により作成されるデータベースは、空です。クエリまたはスクリプトを使用して、必要とされるデータベース構造を作成し、データを追加します。

新規のSQLite データベースの作成方法:

1. データベース接続ウィザードを実行します（データベース接続ウィザードの開始 574 を参照）。
2. **SQLite** を選択して、「**次へ**」をクリックします。



3. [**新規 SQLite データベースの作成**] を選択して、作成されるデータベースファイルの(相対または絶対)パスを入力します（例えば、**c:\users\public\products.sqlite**）。または、[**参照**] をクリックして、フォルダーを選択して、ファイル名テキストボックスにデータベースファイルの名前(例えば、**products.sqlite**)を入力し、[**保存**] をクリックします。

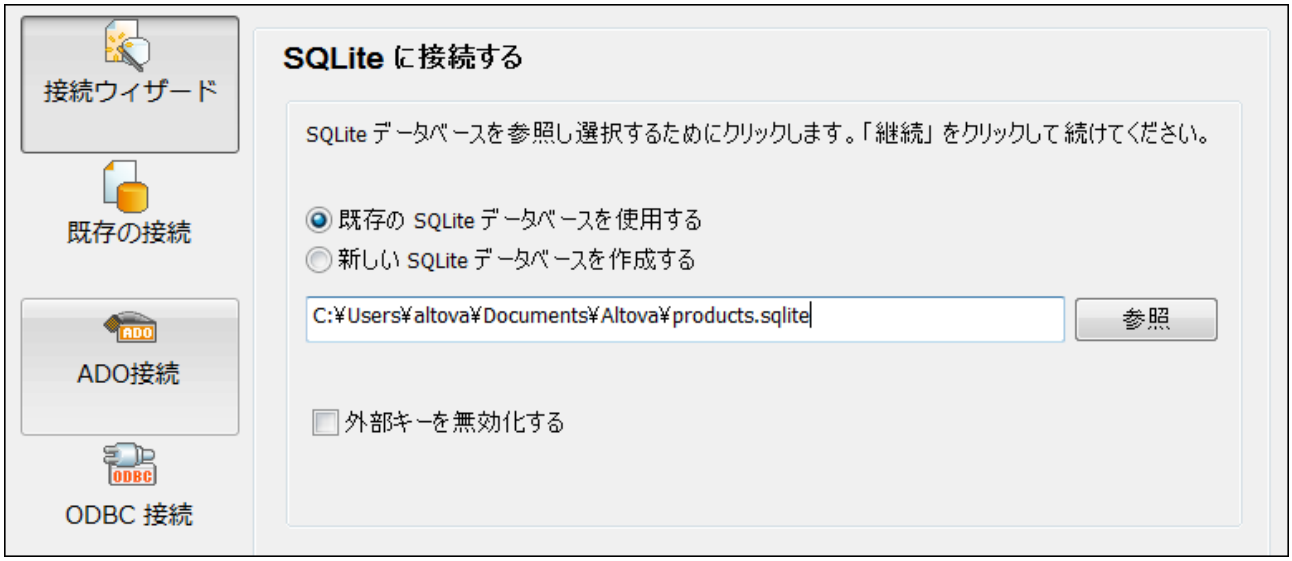
> データベースファイルを作成する場所に対して、書き込みアクセスの許可があることを確認してください。

4. 「**外部キーの無効化**」チェックボックスを任意で選択します。次も参照してください：外部キーの制約 601。
5. 「**接続**」をクリックします。

16.1.8.3 外部キーの制約

XMLSpy, から既存のSQLite データベースに接続する場合、または新規を作成する場合、デフォルトで、外部キーの制約 が有効化されます。外部キーの制約はデータベース内のデータの整合性を保持する手助けを行います。例えば、外部キーが有効化されていると、他のテーブル内に依存関係がある場合、テーブルからレコードを削除することができません。

特定の場合、一時的にこの振る舞いをオーバーライドし、外部キーを無効化する必要がある場合があります。例えば、データ検証エラー無しにデータの列を更新、または、挿入する場合など。SQLite データベースに接続する前に、明示的に外部キーを無効化するには、データベース接続ウィザードの「**外部キーの無効化**」オプションを選択します。



「SQLite への接続」ウィザードページ

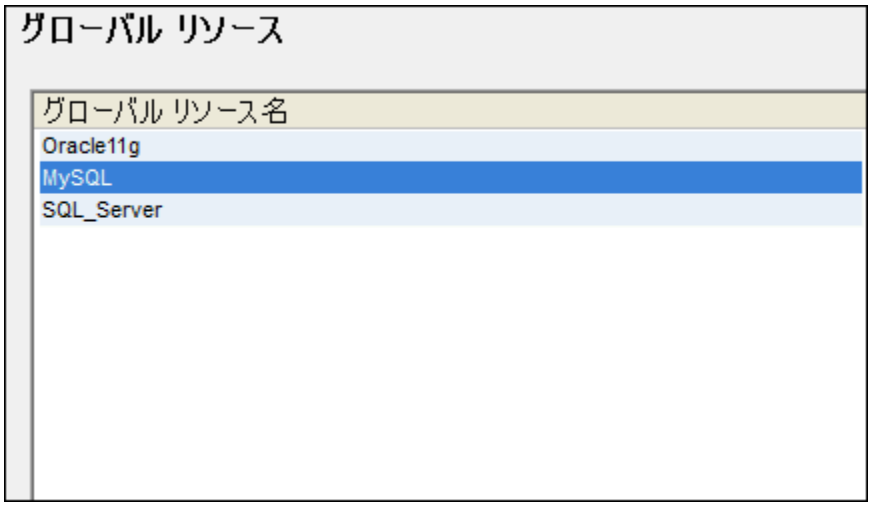
外部キーが無効化されていると、検証チェックのためチェックすることのできなかったデータに対して操作をおこなうことができます。同時に、データベース内に不正確なデータを入れてしまう危険性、または、「無所属の」行を作成する可能性あります。(「無所属の」行の例は、「address」テーブルにリンクされていない「person」テーブル内のアドレスなどが挙げられます。これは、個人名が削除されているにもかかわらず、この個人に関連した住所が削除されていない場合などにこのような行が発生します。)

16.1.9 グローバルリソースからの接続の使用

Altova グローバルリソースはファイル、フォルダー、およびデータベースへのポータブルなレファレンスです。以前にデータベースをグローバルリソースとして既に構成している場合、接続をいつでも再使用することができます(異なる Altova アプリケーション間でも使用可能)。

グローバルリソースからのデータベース接続の使用:

1. データベース接続ウィザードを開始します。 574
2. [グローバルリソース] をクリックします。グローバルリソースとして使用可能なデータベース接続がリストされます。



3. データベース接続履歴から「**接続**」をクリックします。

ヒント: 各グローバルリソースの追加情報を取得するために、マウスカーソルをグローバルリソースにポイントします。

16.1.10 データベース接続のサンプル

ADO、ODBC、またはJDBC を使用したXMLSpy からのデータベースへの接続のためのサンプルがこのセクションには含まれています。ADO.NET 接続サンプルは個別にリストされています サンプルADO.NET 接続文字列 589 を参照してください。For instructions about establishing a native connection to PostgreSQL とSQLite へのネイティブな接続の作成のための手順は、それぞれ PostgreSQL 接続のセットアップ 598 とSQLite 接続のセットアップ 599 を参照してください。

以下の点に注意してください:

- Windows 構成、ネットワーク環境、およびデータベースクライアント、または、サーバーソフトウェアが各サンプルと異なる場合手順は異なります。
- ほぼすべてのデータベースの型では、1つ以上のデータアクセステクノロジー(ADO、ADO.NET、ODBC、JDBC) またはドライバーを使用して接続することができます。データベース接続パフォーマンス、およびドライバーの機能と制限は、XMLSpy 外で構成された選択されたドライバー、(該当する場合)データベースクライアントソフトウェア、および追加接続パラメータなどによって異なります。

16.1.10.1 Firebird (JDBC) への接続

このサンプルではFirebird データベースサーバーへの接続のサンプルの命令について説明されています。

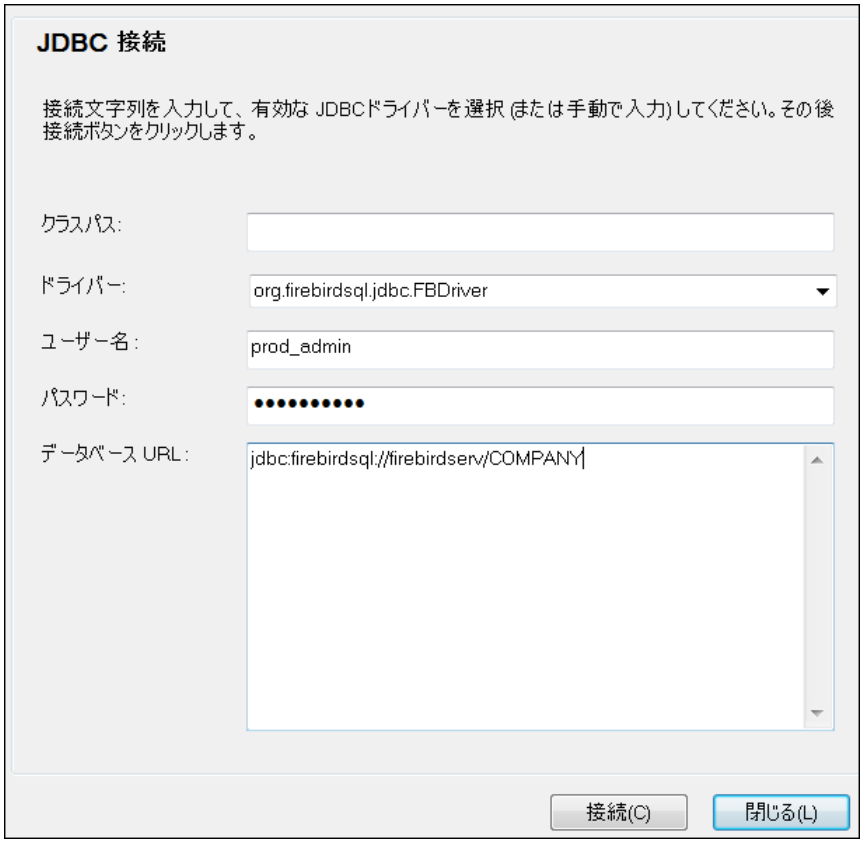
必要条件:

- JRE (Java Runtime Environment) またはJava Development Kit (JDK) がインストールされている必要があります。これはOracle JDK または Oracle OpenJDK などのオープンソースビルドであることができます。XMLSpy は次のロケーションからJava Virtual Machine (JVM) へのパスを次の順序で決定します: a) アプリケーションオプション内で設定したカスタムJVM パス b) Windows レジストリ内で見つかったJVM パス。c) `JAVA_HOME` 環境変数。
- XMLSpy (32-ビット、64-ビット) のプラットフォームがJRE/JDK のプラットフォームに一致することを確認してください。

- Firebird JDBC ドライバーが使用中のオペレーティングシステムで使用可能である必要があります（データベースへの接続を可能にする .jar ファイルのフォームをとります）。ドライバーは Firebird Web サイトからダウンロードすることができます（ http://www.firebirdsql.org/ ）。このサンプルは、*Jaybird 2.2.8* を使用します。
- データベース接続の詳細があります: ホスト、データベース パス、またはエイリアス、ユーザー名、および パスワード。

JDBC を使用して Firebird に接続する

1. データベース接続ウィザードを開始します 574.
2. 「JDBC 接続」をクリックします。
3. 「クラス パス」の横にデータベースへの接続を与える .jar ファイルへの パスを入力します。必要であれば、セミコロンで区切られた .jar ファイル パスのリストを入力することもできます。このサンプルでは、必要とされる .jar ファイルは次の パスを使用して見つけることができます: **C:\jdbc\firebird\jaybird-full-2.2.8.jar**。.jar ファイル パスをオペレーティングシステムの CLASSPATH 環境変数に追加した場合、「クラス パス」テキスト ボックスを空白のままにすることができます。（次も参照: CLASSPATH の構成 596）。
4. 「ドライバー」ボックスから、**org.firebirdsql.jdbc.FBDriver** を選択します。このエントリは、有効な .jar ファイル パスが、「クラス パス」テキスト ボックス内、または、オペレーティングシステムの CLASSPATH 環境変数内で検出された場合のみ使用することができます（前のステップを参照してください）。



5. 対応するテキスト ボックスに、データベースへのユーザー名と パスワードを入力します。
6. URL テキスト ボックスに、データベースサーバーに適用することのできる ハイライトされた値を置き換える、データベースサーバーの接続文字列を入力します。

```
jdbc:firebirdsql://<host>[:<port>]/<databese path or alias>
```

7. 「接続」をクリックします。

16.1.10.2 Firebird (ODBC) への接続

このサンプルでは、Linux サーバー上で実行される、Firebird 2.5.4 データベースへの接続について説明されています。

必要条件:

- Firebird データベースサーバーは、クライアントからのTCP/IP 接続を受け入れるように構成されています。
- Firebird ODBC ドライバーを使用中のオペレーティングシステムにインストールする必要があります。このサンプルは、を使用します。 Firebird ODBC ドライバー バージョン 2.0.3.154 は、Firebird ウェブサイト (http://www.firebirdsql.org/) からダウンロードすることができます。
- Firebird クライアントを、使用中のオペレーティングシステムにインストールする必要があります。 Firebird 2.5.4 クライアントのためのスタンドアロンのインストーラーが存在しない点に注意してください。また、クライアント は、サーバーのインストール パックの一部です。Firebird サーバー インストール パックをFirebird ウェブサイト(http://www.firebirdsql.org/) からダウンロードすることができます。ダウンロードの際、[スーパークラシック/クラシック または スーパーサーバーのための、Windows 実行ファイル] インストーラーを選択してください。クライアントファイルのみをインストールする場合、[**最小クライアント インストール- サーバーツール無し**] を選択します。

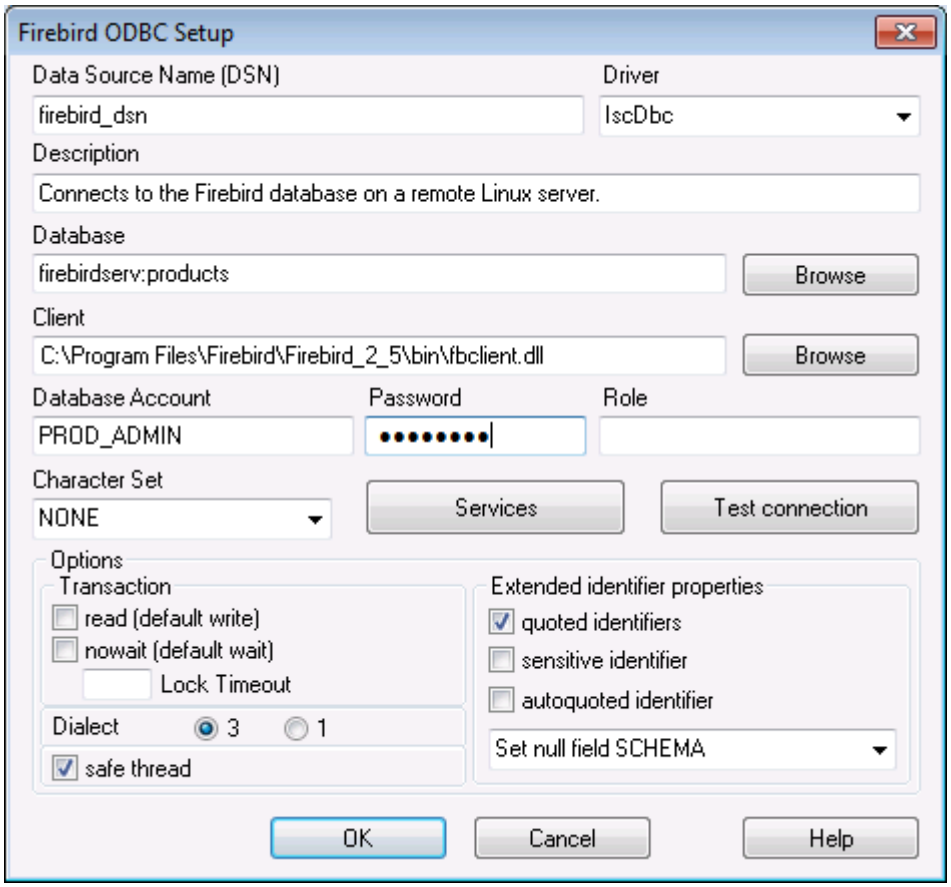
> **重要点:**
>
> - Firebird ODBC ドライバーおよび クライアント (32 ビット または 64 ビット) のプラットフォームは、XMLSpy のプラットフォームに適応する必要があります。
> - Firebird クライアントのバージョンは、接続されているFirebird サーバーのバージョンに適応する必要があります。

- データベース接続の詳細 サーバー上のサーバーホスト名 またはIP アドレス、データベース パス(または エイリアス)、ユーザー名、および パスワード。

ODBC を使用して Firebird に接続する:

1. データベース接続ウィザードを開始します[574]。
2. [**ODBC 接続**] をクリックする。
3. [**ユーザー DSN**] を選択して、(または、管理者の特権がある場合、 [**システム DSN**] を選択して、)[**追加**] をクリックします。

4. Firebird ドライバーを選択して、をクリックします。そして [ユーザー DSN] (前のステップでの選択により、または、[システム DSN])。Firebird ドライバーがリスト内に存在しない場合、使用中のオペレーティングシステムにインストールされていることを確認してください (使用可能な ODBC ドライバーの表示 593 を参照)。



5. データベース接続の詳細を入力します:

データソース名 (DSN)	作成する詳細名とデータソースを入力します。

データベース	サーバーホスト名 または IP アドレスをコロンとデータベース エイリアス(または パス)の後に入力します。このサンプルでは、ホスト名は、`firebirdserv`、データベースエイリアスは、`products` です:
	
	`firebirdserv:products`
	
	データベース エイリアスを使用することにより、サーバー側上では、データベース管理者は、エイリアス製品を構成し、サーバー上の実際の Firebird (.fdb) データベースファイルをポイントします (詳細は、Firebird のドキュメントを参照してください)。
	
	ホスト名の代わりに サーバー IP アドレス、エイリアスの代わりに パス を使用することもできます。ですから、以下のサンプル接続の文字列は有効です:
	
	`firebirdserver:/var/Firebird/databases/butterflies.fdb`
	`127.0.0.1:D:\Misc\Lenders.fdb`
	データベースが Windows マシン上に存在する場合、[参照] をクリックして、Firebird (.fdb) データベースファイルを直接選択します。

(C) 2015-2021 Altova GmbH

クライアント	**fbclient.dll** ファイルへの パスを入力します。デフォルトでは、これはFirebird インストールディレクトリのサブディレクトリです。
データベース アカウント	データベース管理者（このサンプルの場合 PROD_ADMIN）により与えられたデータベースユーザー名を入力します。
パスワード	データベース管理者により与えられた データベース パスワード を入力します。

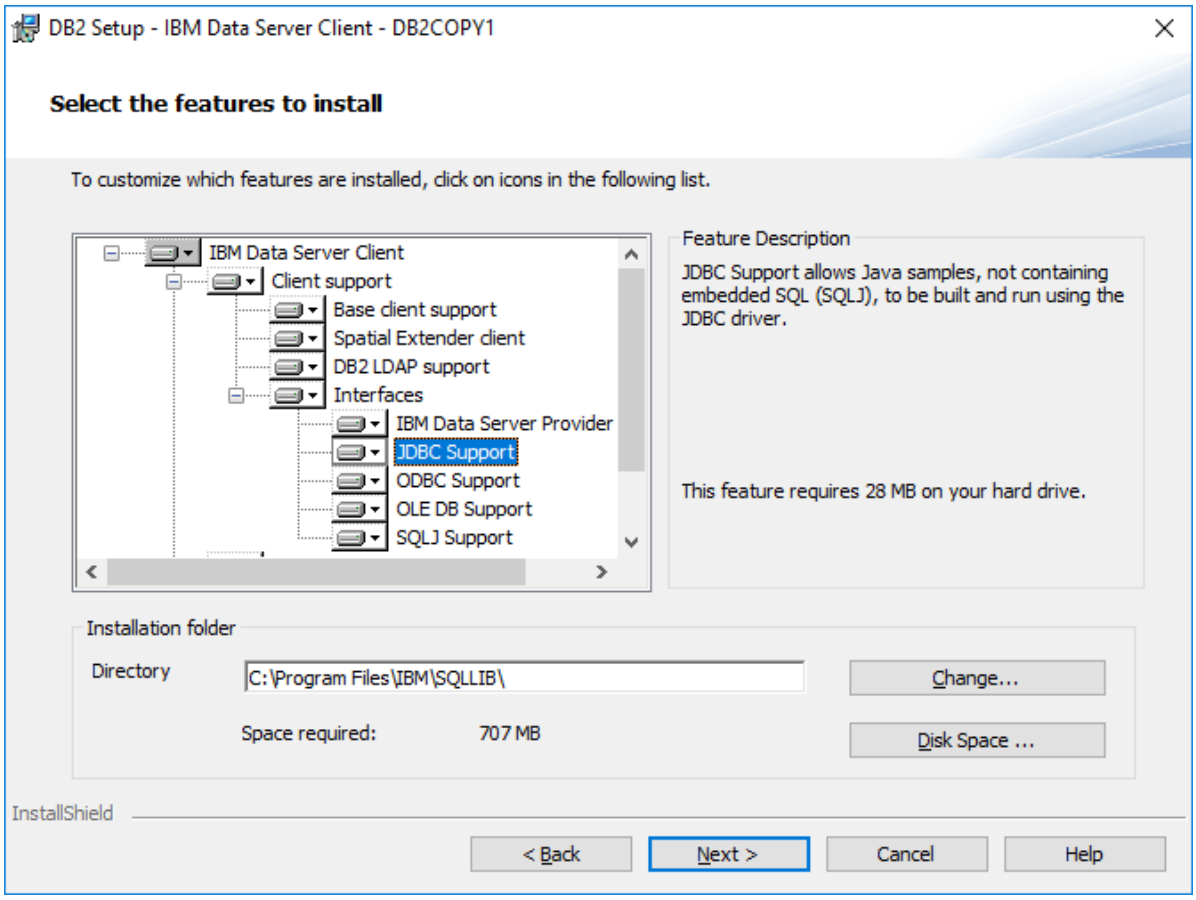
6. 「OK」をクリックします。

16.1.10.3 IBM DB2 (JDBC) への接続

このサンプルでは、JDBC を使用して **IBM DB2** データベースサーバーに接続する手順のサンプルについて説明されています。

必要条件:

- JRE（Java Runtime Environment）またはJava Development Kit（JDK）がインストールされている必要があります。Oracle JDK またはOracle OpenJDK などのオープンソースビルドであることができます。XMLSpy は次のロケーションから Java Virtual Machine（JVM）へのパスを次の順序で決定します: a）アプリケーション オプション内で設定したカスタム JVM パス b）Windows レジストリ内で見つかった JVM パス。c）JAVA_HOME 環境変数。
- XMLSpy（32-ビット、64-ビット）のプラットフォームがJRE/JDK のプラットフォームに一致することを確認してください。このサンプルはOracle のOpenJDK 11.0 64-bit を使用し、その結果 XMLSpy の64 ビットバージョンを使用します。
- （データベースへの接続を与える1 つ、または、複数の.jar ファイル） JDBC ドライバーが使用中のオペレーティングシステムで使用可能である必要があります。**IBM Data Server Client** バージョン 10.1（64-ビット）をインストール後使用することのできる JDBC ドライバーをこのサンプルは使用しています。JDBC ドライバーをインストールするためには、**通常** インストールを選択してください。または、インストールウィザードでこのオプションを明示的に選択してください。

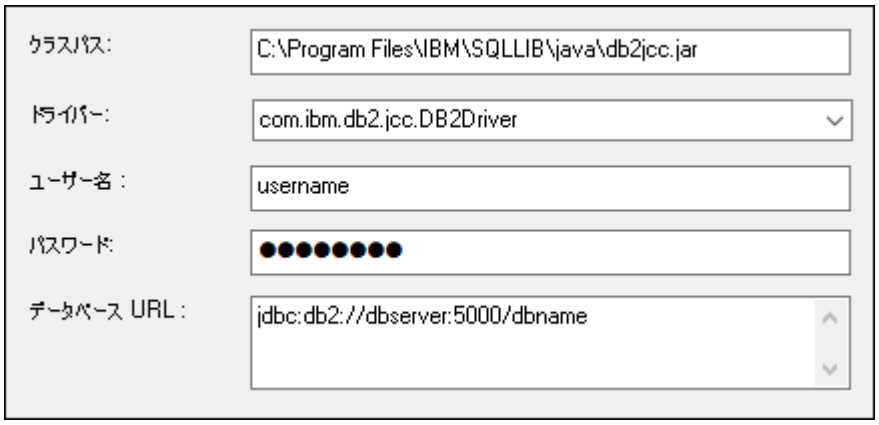


デフォルトのインストール パスを変更していない場合、インストール後必要とされる .jar ファイルは **C:\Program Files\IBM\SQLLIB\java** ディレクトリ内で見つけることができます。

- 以下のデータベース詳細が必要になります: ホスト、データベース、ポート、ユーザー名、および、パスワード。

JDBC を使用して IBM DB2 に接続する:

1. データベース接続ウィザードを開始します[574].
2. 「JDBC 接続」をクリックします。
3. 「クラス パス」の横に、データベースへの接続を与える .jar ファイルへの パスを入力します。このサンプルは **C:\Program Files\IBM\SQLLIB\java\db2jcc.jar** を参照します。データベースサーバーバージョンにより **db2jcc4.jar** ドライバーを参照する必要がある場合があります。ドライバーの互換性に関しては IBM ドキュメントを参照してください（http://www-01.ibm.com/support/docview.wss?uid=swg21363866）。.jar ファイル パスをオペレーティングシステムの CLASSPATH 環境変数に追加している場合クラス パステキストボックスを空にしておくことができます（CLASSPATH の構成[596]を参照してください）。
4. 「ドライバー」ボックスから、**com.ibm.db2.jcc.DB2Driver** を選択します。このエントリは、有効な .jar ファイル パスが、「クラス パス」テキストボックス内、または、オペレーティングシステムの CLASSPATH 環境変数内で検出された場合のみ使用することができます（前のステップを参照してください）。



5. データベースユーザーのユーザー名とパスワードを対応するテキストボックスに入力します。
6. JDBC 接続文字列をデータベース URL テキストボックスに入力します。使用中のデータベースサーバーに適用することのできるものと接続の詳細が置き換えられていることを確認してください。

```
jdbc:db2://hostName:port/databaseName
```

7. 「接続」をクリックします。

16.1.10.4 IBM DB2 (ODBC) への接続

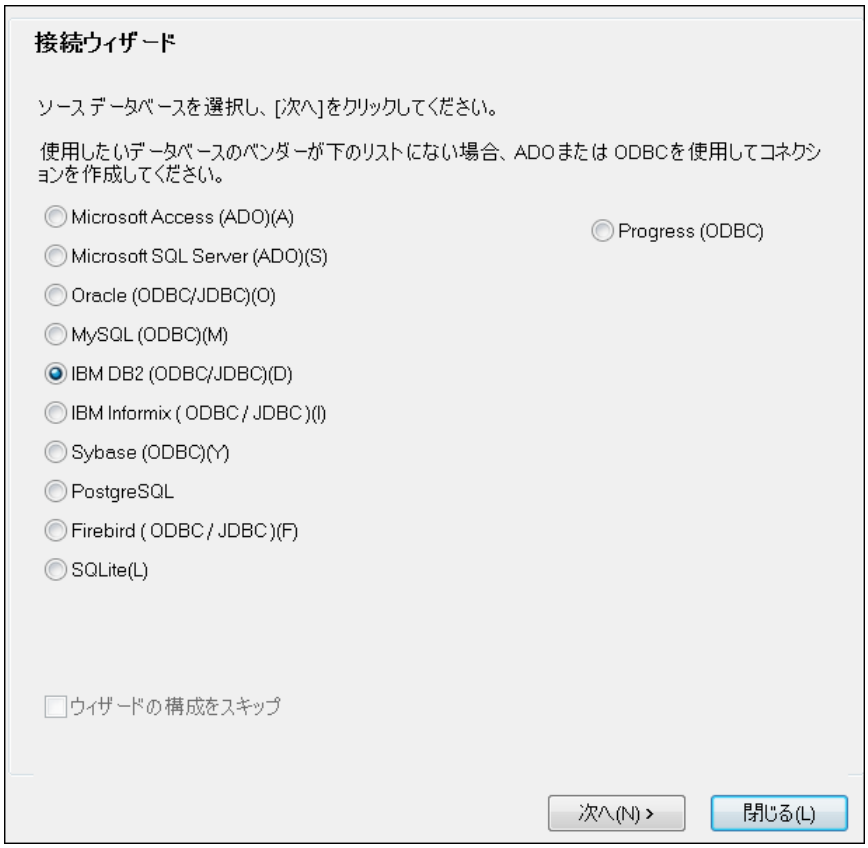
このトピックではODBC を使用した、IBM DB2 への接続のサンプル手順について説明されています。

前提条件:

- IBM Data Server Client は、使用中のオペレーティングシステムにインストールおよび構成される必要があります(このサンプルの場合はIBM Data Server Client 9.7 を使用します)。インストールの手順に関しては、IBM DB2 ソフトウェアにより提供されたドキュメントを確認してください。 IBM Data Server Client インストール後、使用中のコンピューターで ODBC ドライバーが使用可能か確認してください(使用可能な ODBC ドライバーの表示 593 を参照)。
- データベースエイリアスを作成します。複数の作成方法があります:
  - IBM DB2 構成アシスタントから
  - IBM DB2 コマンドラインプロセッサーから
  - ODBC データソースウィザードから(この場合は、下に表示される手順を参照してください)
- データベース接続の詳細: ホスト、データベース、ユーザー名およびパスワード

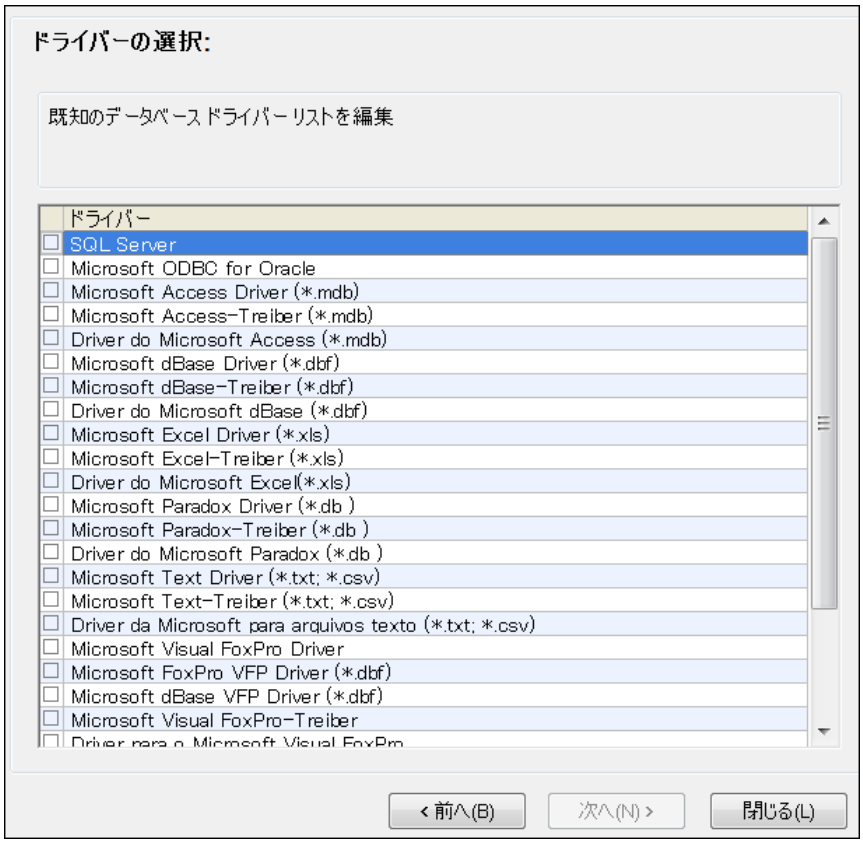
IBM DB2 への接続:

1. データベース接続ウィザードを開始します 574 。**IBM DB2 (ODBC/JDBC)** を選択します。

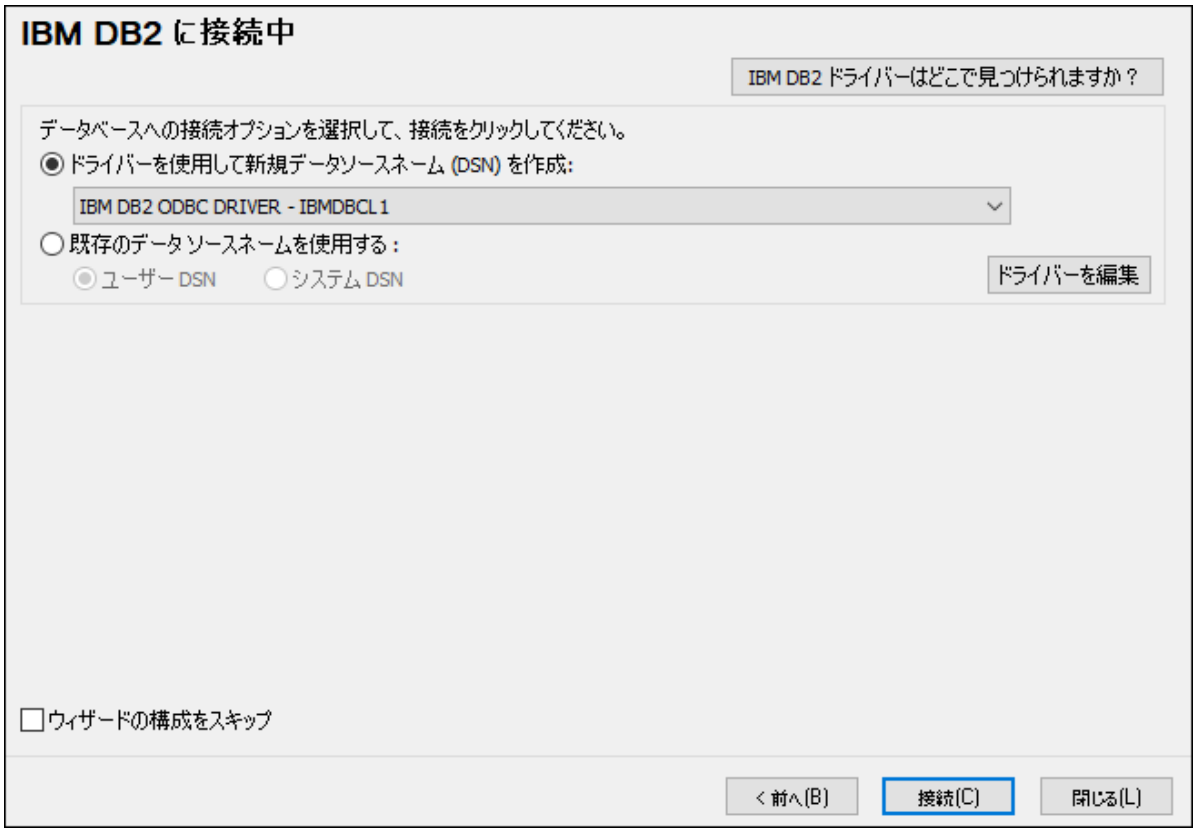


2. 「次へ」をクリックします。

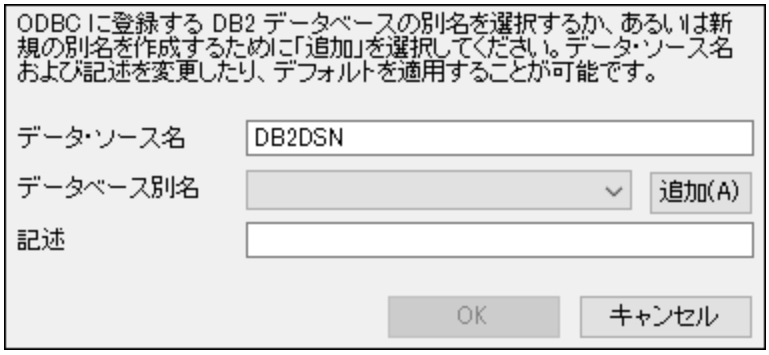
3. **ODBC** を選択して「次へ」をクリックします。データベースのための既知のリストを編集するようにプロンプトされると、IBM DB2 に適用することのできるドライバーを選択して(前提条件 609 を参照)、「次へ」をクリックします。



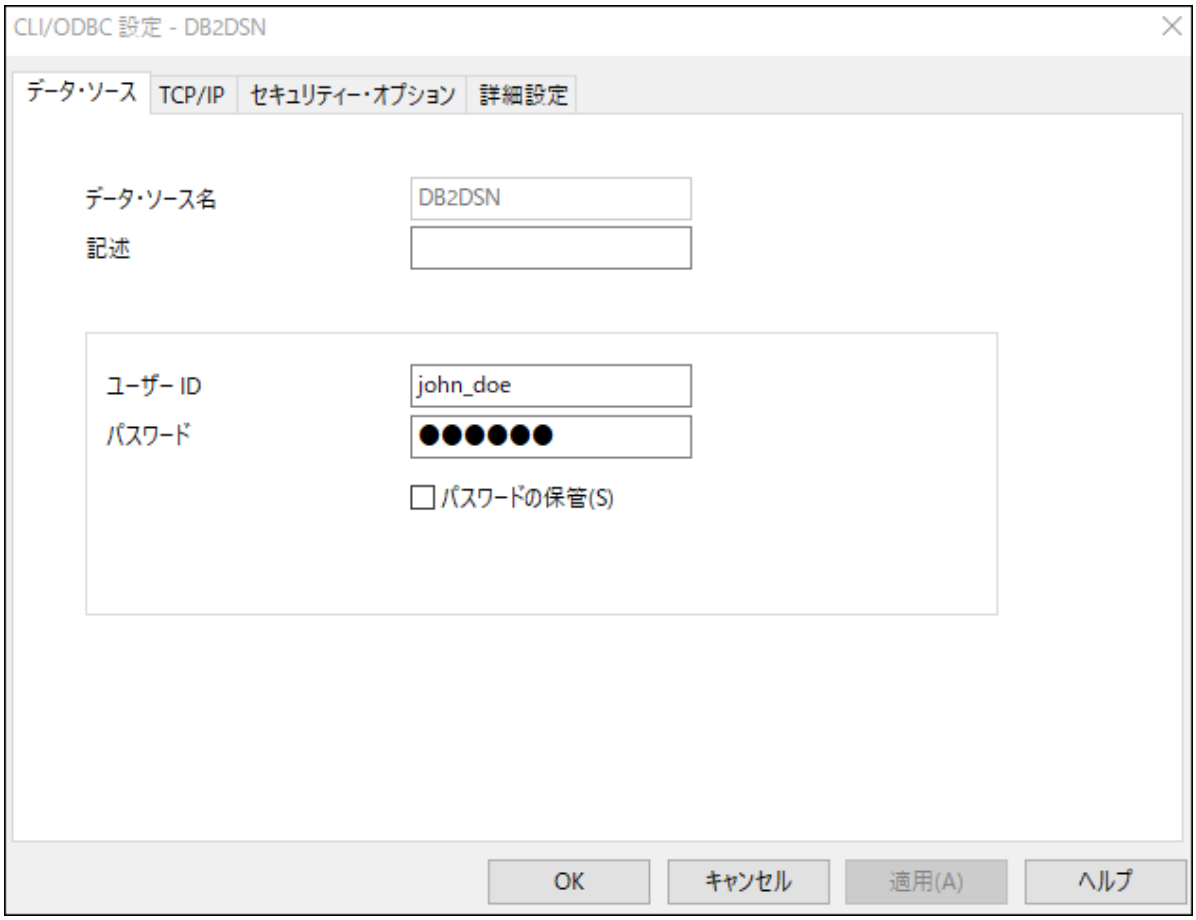
4. リストからIBM DB2 を選択して、「**接続**」をクリックします。（ドライバーで使用可能なリストを編集するには、[**ドライバーの編集**] をクリックして、使用可能なドライバーのリストを編集します。クリックして追加または削除するIBM DB2 ドライバーをチェックまたはチェックの解除をします。ドライバーを選択または選択の解除をします。）



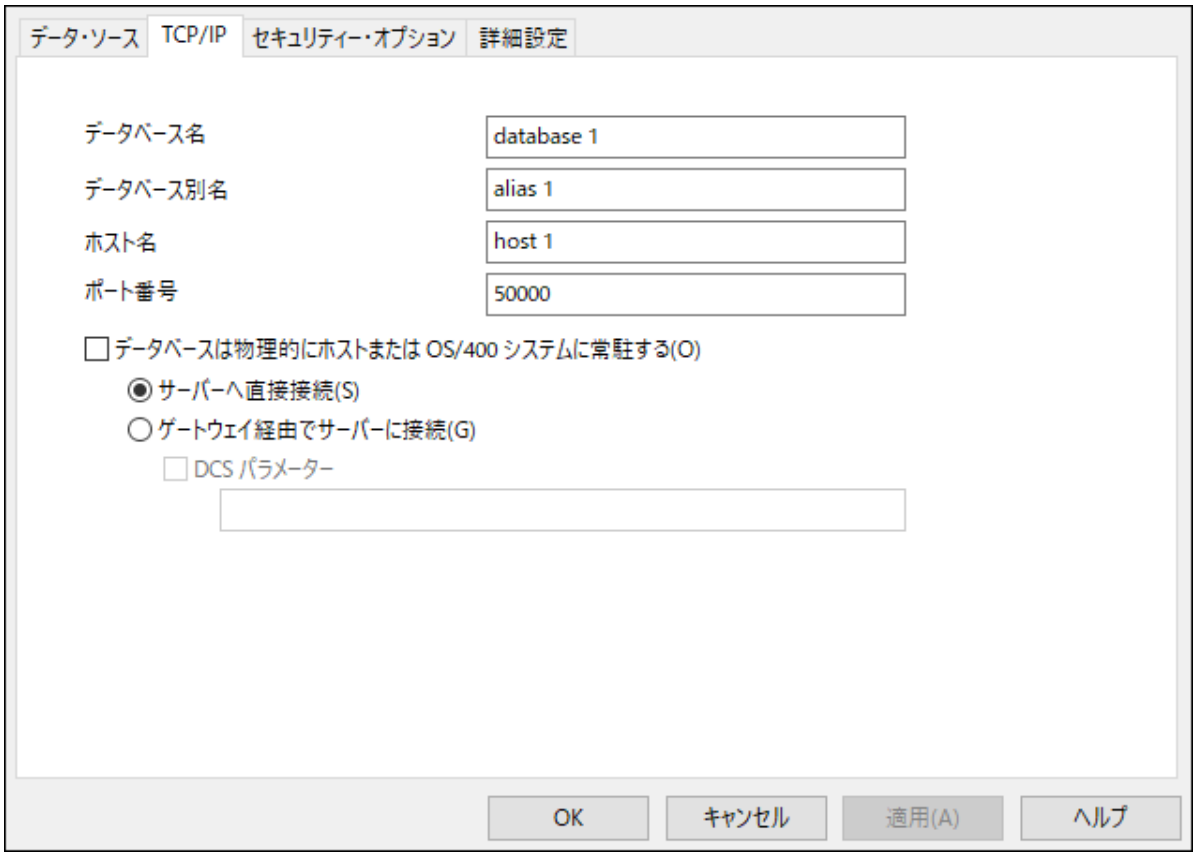
5. ソース名を入力します（このサンプルでは**DB2DSN**）「追加」をクリックします。



6. [データソース] タブで、データベースのユーザー名とパスワードを入力します。



7. **[TCP/IP]** タブで、データベース名、エイリアスの名前、ホスト名とポート名を入力して「OK」をクリックします。



8. ユーザ名とパスワードを再入力して「OK」をクリックします。

16.1.10.5 IBM DB2 for i (JDBC) への接続

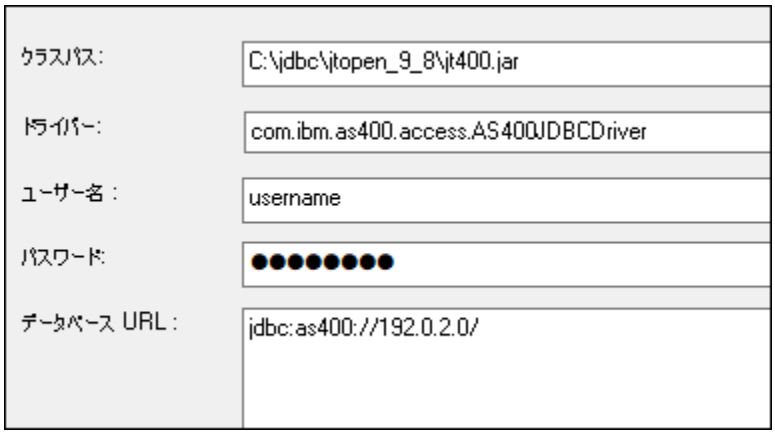
このサンプルでは JDBC を介して **IBM DB2 for i** データベースサーバーに接続する方法について説明されています。

必要条件

- JRE (Java Runtime Environment) または Java Development Kit (JDK) がインストールされている必要があります。これは Oracle JDK または Oracle OpenJDK などのオープンソースビルドであることができます。XMLSpy は次のロケーションから Java Virtual Machine (JVM) へのパスを次の順序で決定します: a) アプリケーション オプション内で設定したカスタム JVM パス b) Windows レジストリ内で見つかった JVM パス。c) `JAVA_HOME` 環境変数。
- XMLSpy (32-ビット、64-ビット) のプラットフォームが JRE/JDK のプラットフォームに一致することを確認してください。このサンプルは Oracle の OpenJDK 11.0 64-bit を使用し、その結果 XMLSpy の 64 ビット バージョンを使用します。
- JDBC ドライバー (データベースへの接続を提供する 1 つ、または、複数の .jar ファイル) が使用中のオペレーティングシステム上で使用可能である必要があります。このサンプルは以下のオープンソースを使用します: **Toolbox for Java/JTOpen** バージョン 9.8 (http://jt400.sourceforge.net/)。パッケージをダウンロードし、ローカルディレクトリにアンパックすると、必要な .jar ファイルが **lib** サブディレクトリで使用できるようになります。
- 以下のデータベース接続の詳細が必要です: ホスト、ユーザー名、および パスワード

JDBC を使用して IBM DB2 for i に接続する方法:

1. データベース接続ウィザードを開始します 574。
2. 「**JDBC 接続**」をクリックします。
3. 「クラス パス」の横にデータベースへの接続を与える .jar ファイルへのパスを入力します。必要であれば、セミコロンで区切られた .jar ファイル パスのリストを入力することもできます。このサンプルでは、必要とされる .jar ファイルは次のパスを使用して見つけることができます: **C:\Informix_JDBC_Driver\lib\ifxjdbc.jar**。.jar ファイル パスをオペレーティングシステムの CLASSPATH 環境変数に追加した場合、「クラス パス」テキスト ボックスを空白のままにすることができます。(次も参照: CLASSPATH の構成 596)。
4. 「ドライバー」ボックスから、**com.ibm.as400.access.AS400JDBCDriver** を選択します。このエントリは、有効な .jar ファイル パスが、「クラス パス」テキスト ボックス内、または、オペレーティングシステムの CLASSPATH 環境変数内で検出された場合のみ使用することができます (前のステップを参照してください)。



5. データベースユーザーのユーザー名と パスワードを対応するテキスト ボックスに入力します。
6. JDBC 接続文字列をデータベース URL テキスト ボックスに入力します。`host` をデータベースサーバーのホスト名、または、IP アドレスと置き換えてください。

```
jdbc:as400://host
```

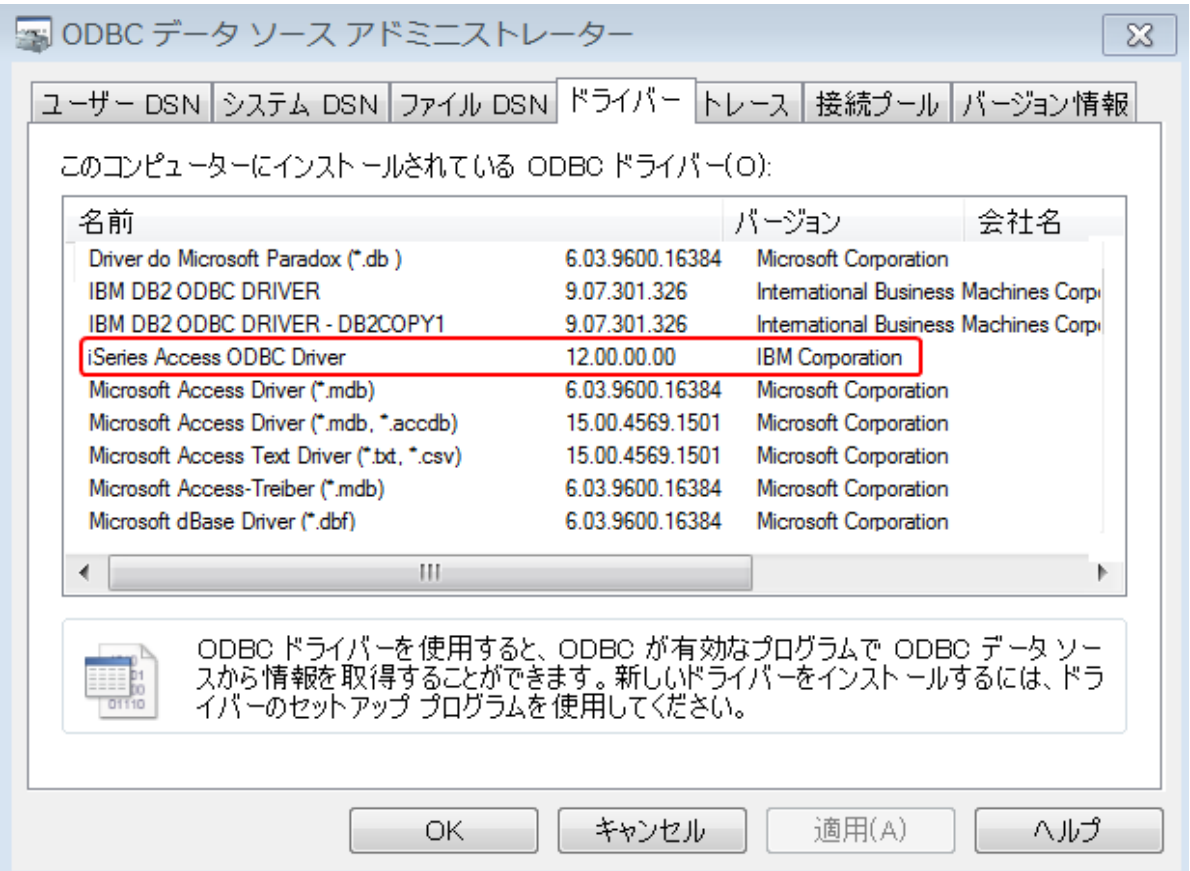
7. 「接続」をクリックします。

16.1.10.6 IBM DB2 for i (ODBC) への接続

このサンプルでは、ODBC を使用して *IBM DB2 for i* データベースへの接続について説明されています。

前提条件:

- *IBM System i Access for Windows* は使用中のオペレーティングシステム（このサンプルでは *IBM System i Access for Windows V6R1M0* を使用しています）にインストールされる必要があります。インストールの手順は、*IBM DB2 for i* ソフトウェアにより提供されているドキュメントを参照してください。インストール後、ODBC ドライバーが使用中のコンピューターで使用可能か確認してください（使用可能なODBC ドライバーの表示 593 を参照）。

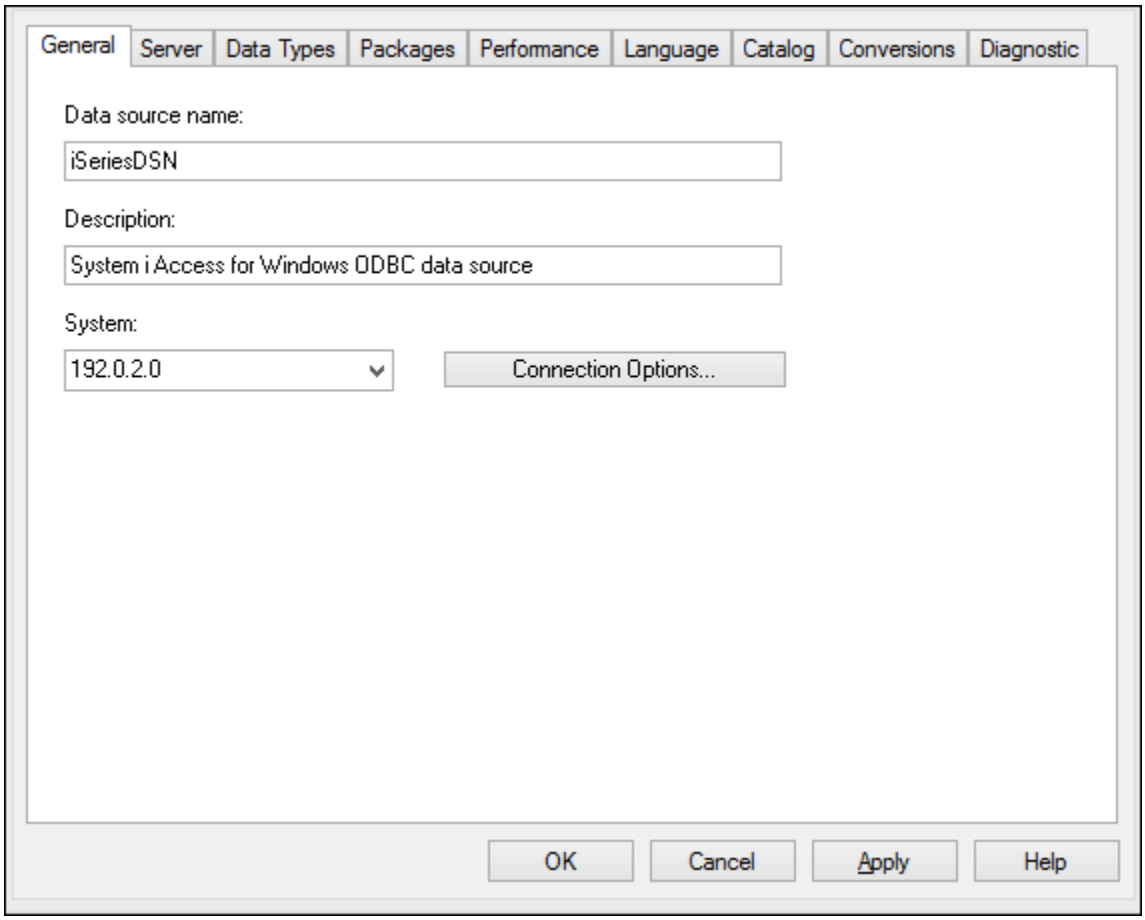


- データベース接続の詳細: データベースサーバーの I.P. アドレス、データベースユーザー名、および パスワード
- *i Navigator* を実行して、新規接続を作成するウィザードの手順を踏んで新規接続を作成してください。システムの指定をプロンプトされると、データベースサーバーの I.P. アドレスを入力してください。接続を作成した後、検証する（接続をクリックして、**ファイル > 診断 > 接続の検証** を選択）ことが奨励されます。接続エラーが発生した場合、データベースサーバー管理者に連絡してください。

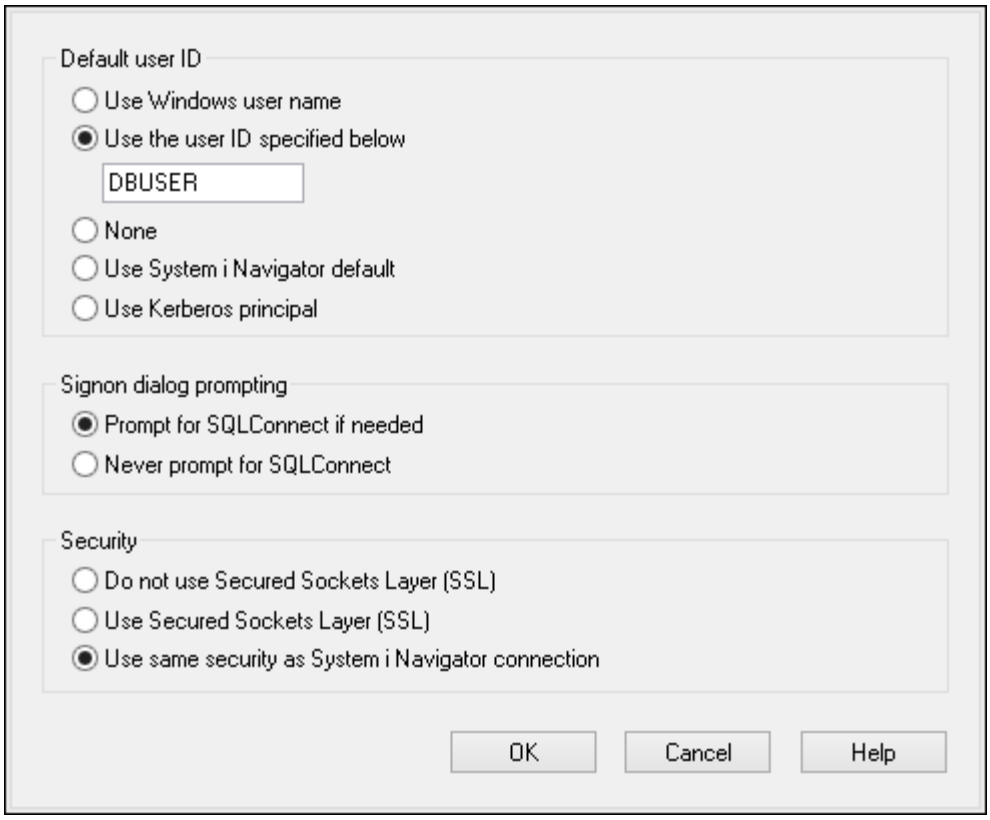
IBM DB2 for i への接続:

1. データベース接続ウィザードを開始します 574.
2. [ODBC 接続] をクリックします。
3. [ユーザー DSN] をクリックします。(または、[システム DSN] または、後に続く手順が同じ [ファイル DSN] をクリックします)。
4. 「追加」 をクリックします。
5. **[iSeries Access ODBC Driver]** をリストから選択します。[ユーザー DSN] (または、該当する場合 [システム DSN]) をクリックします。

6. データソース名を入力して、システムコンボボックスから接続を選択します。このサンプルでは、データソース名は **iSeriesDSN** でシステムは **192.0.2.0** です。



7. [接続オプション] をクリックします。[指定されたユーザー ID を使用] を選択して、データベースユーザー（このサンプル内では **DBUSER**）の名前を入力します。



8. 「OK」をクリックします。DSN 内のリストで新しいデータソースを使用することができます。
9. 「接続」をクリックします。
10. データベースのユーザー名とパスワードをプロンプトされると入力して「OK」をクリックします。

16.1.10.7 IBM Informix (JDBC) への接続

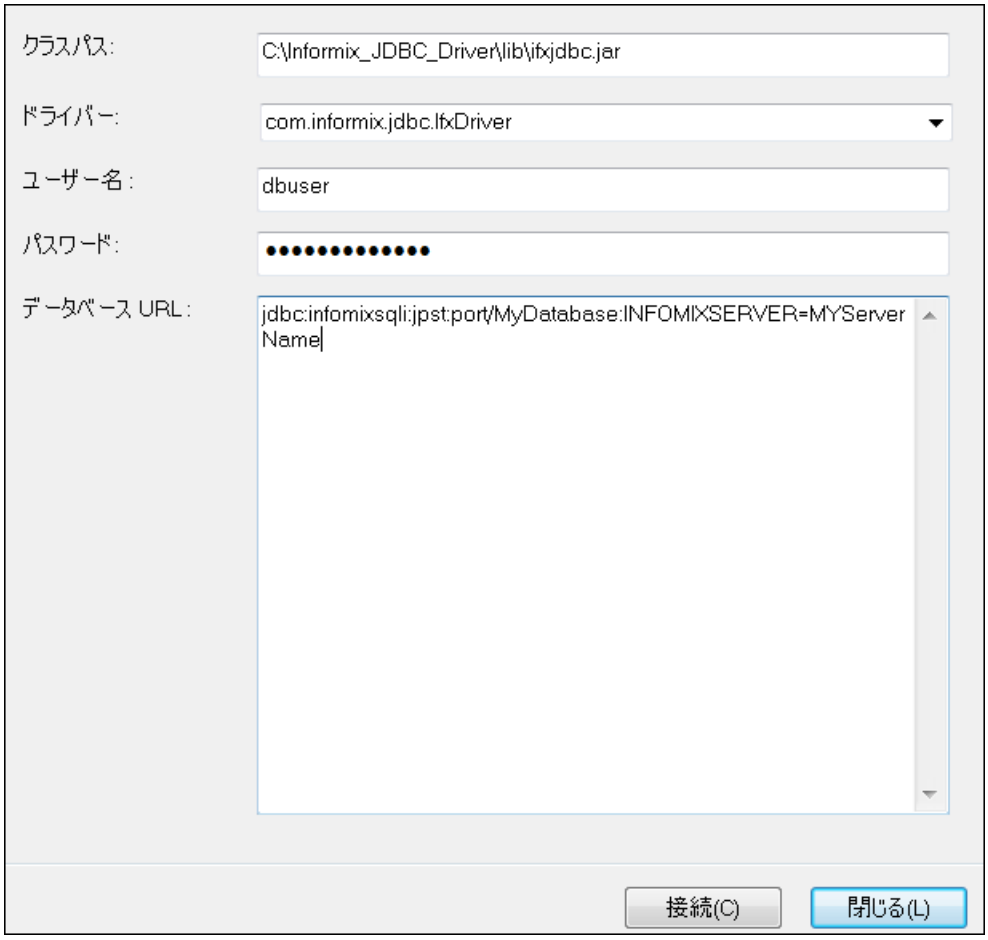
このサンプルでは、JDBC を使用してIBM Informix データベースサーバーに接続する手順のサンプルについて説明されています。

前提条件:

- JRE (Java Runtime Environment) またはJava Development Kit (JDK) がインストールされている必要があります。これはOracle JDK または Oracle OpenJDK などのオープンソースビルドであることができます。XMLSpy は次のロケーションからJava Virtual Machine (JVM) へのパスを次の順序で決定します: a) アプリケーション オプション内で設定したカスタムJVM パス b) Windows レジストリ内で見つかったJVM パス。c) `JAVA_HOME` 環境変数。
- XMLSpy (32-ビット、64-ビット) のプラットフォームがJRE/JDK のプラットフォームに一致することを確認してください。
- (データベースへの接続を与える1 つ、または、複数の.jar ファイル) JDBC ドライバーが使用中のオペレーティングシステムで使用可能である必要があります。この例では、IBM Informix JDBC ドライバー バージョン3.70 が使用されています。ドライバーのインストールの説明に関しては、ドライバーに付随するドキュメント、または、「IBM Informix JDBC ドライバプログラマーズガイド」を参照してください)。
- データベース接続の詳細: ホスト、サーバーの名前、データベース、ポート ユーザー名、およびパスワード

JDBC を使用して IBM Informix に接続する

1. データベース接続ウィザードを開始します[574].
2. 「JDBC 接続」をクリックします。
3. 「クラスパス」の横にデータベースへの接続を与える .jar ファイルへのパスを入力します。必要であれば、セミコロンで区切られた .jar ファイルパスのリストを入力することもできます。このサンプルでは、必要とされる .jar ファイルは次のパスを使用して見つけることができます: **C:\Informix_JDBC_Driver\lib\ifxjdbc.jar**。.jar ファイルパスをオペレーティングシステムの CLASSPATH 環境変数に追加した場合、「クラスパス」テキストボックスを空白のままにすることができます。(次も参照: CLASSPATH の構成[596])。
4. 「ドライバー」ボックスから、**com.informix.jdbc.IfxDriver** を選択します。このエントリは、有効な .jar ファイルパスが、「クラスパス」テキストボックス内、または、オペレーティングシステムの CLASSPATH 環境変数内で検出された場合のみ使用することができます(前のステップを参照してください)。



5. データベースへのユーザー名とパスワードを対応するテキストボックスに入力します。
6. ハイライトされた値を、使用するデータベースサーバーに適用する値と置き換えて、データベース URL テキストボックス内にデータベースサーバーへの接続文字列を入力します。

```
jdbc:informix-sqli://hostName:port/databaseName:INFORMIXSERVER=myserver;
```

7. 「接続」をクリックします。

16.1.10.8 MariaDB (ODBC) への接続

このサンプルでは、ODBC を使用したMariaDB データベースへの接続方法について説明されています。

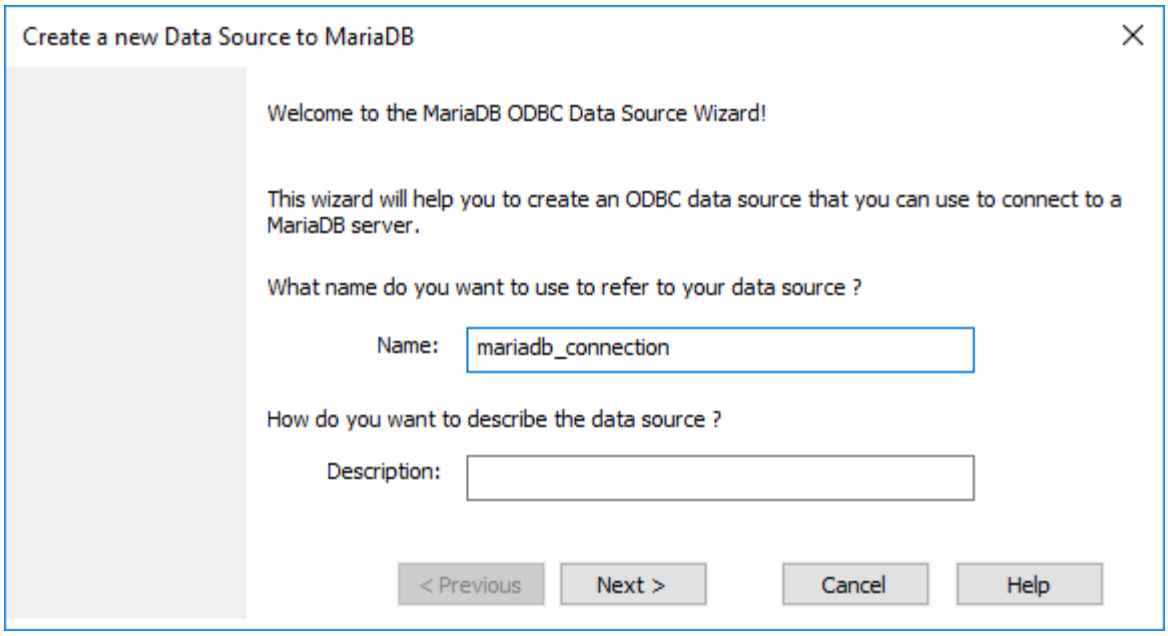
前提条件:

- MariaDB コネクタ/ODBC (https://downloads.mariadb.org/connector-odbc/) がインストールされている必要があります。
- 次のデータベース接続詳細が存在します: ホスト、データベース、ポート、ユーザー名、および、パスワード。

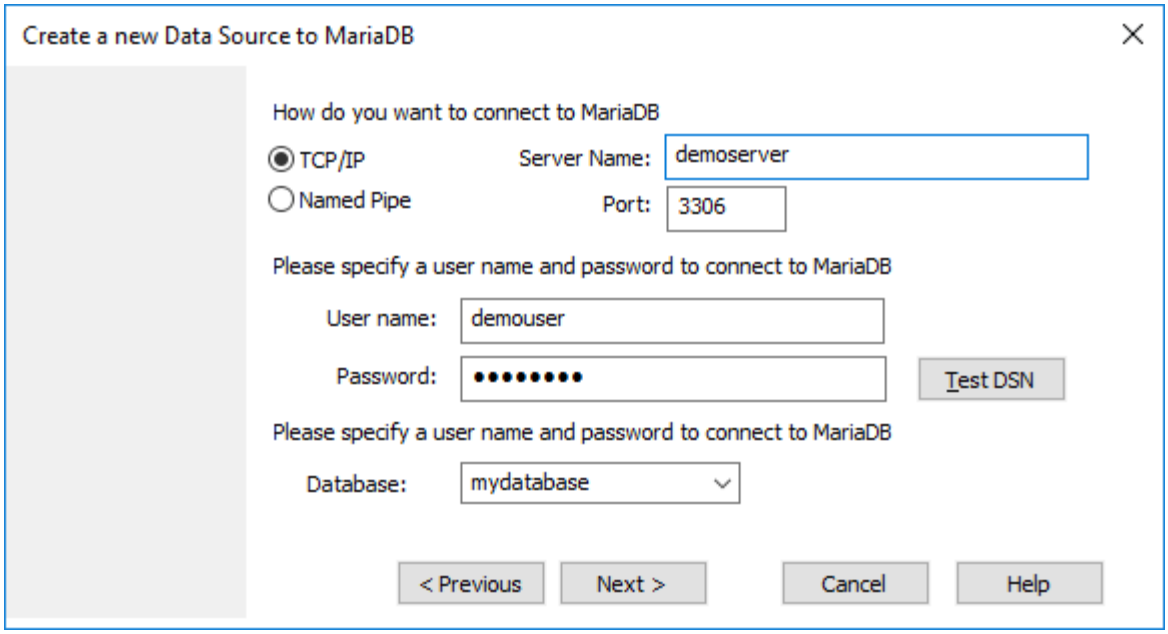
ODBC を使用した MariaDB への接続の方法:

1. データベース接続ウィザードを開始します 574 。
2. 「MariaDB (ODBC)」を選択して、「次へ」をクリックします。

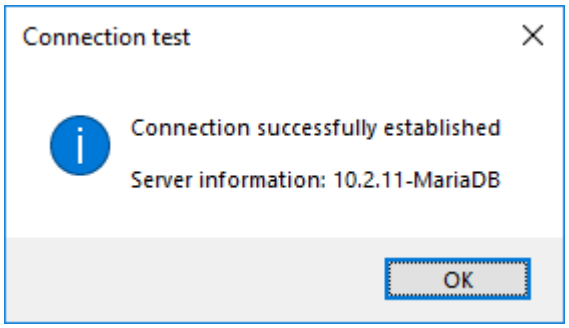
3. 「ドライバーを使用して新規のデータソース名 (DSN) を作成する」を選択し、「MariaDB ODBC 3.0 Driver」を選択します。このようなドライバーがリスト内で使用できない場合、「ドライバーの編集」をクリックし、使用することのできるMariaDB ドライバーを選択します (リストには使用中のオペレーティングシステム上にインストールされているすべてのODBC ドライバーが含まれています)。
4. 「接続」をクックします。



5. 名前を入力し、任意で、このODBC データソースを将来識別できる詳細を与えます。



6. データベース接続資格情報(TCP/IP サーバー、ユーザー、パスワード)を入力し、データベースを選択して、「Test DSN」をクリックします。接続に成功すると、メッセージが表示されます:



7. 「次へ」をクリックしてウィザードを完了します。他のパラメータが必要になる場合がありますが(例えば、安全な接続を使用して、MariaDB に接続する場合は、SSL 証明書が必要になります)。

メモ　データベースサーバーがリモートの場合、サーバー管理者が遠隔の接続を使用中のマシンの IP アドレスから受け入れるように構成する必要があります。

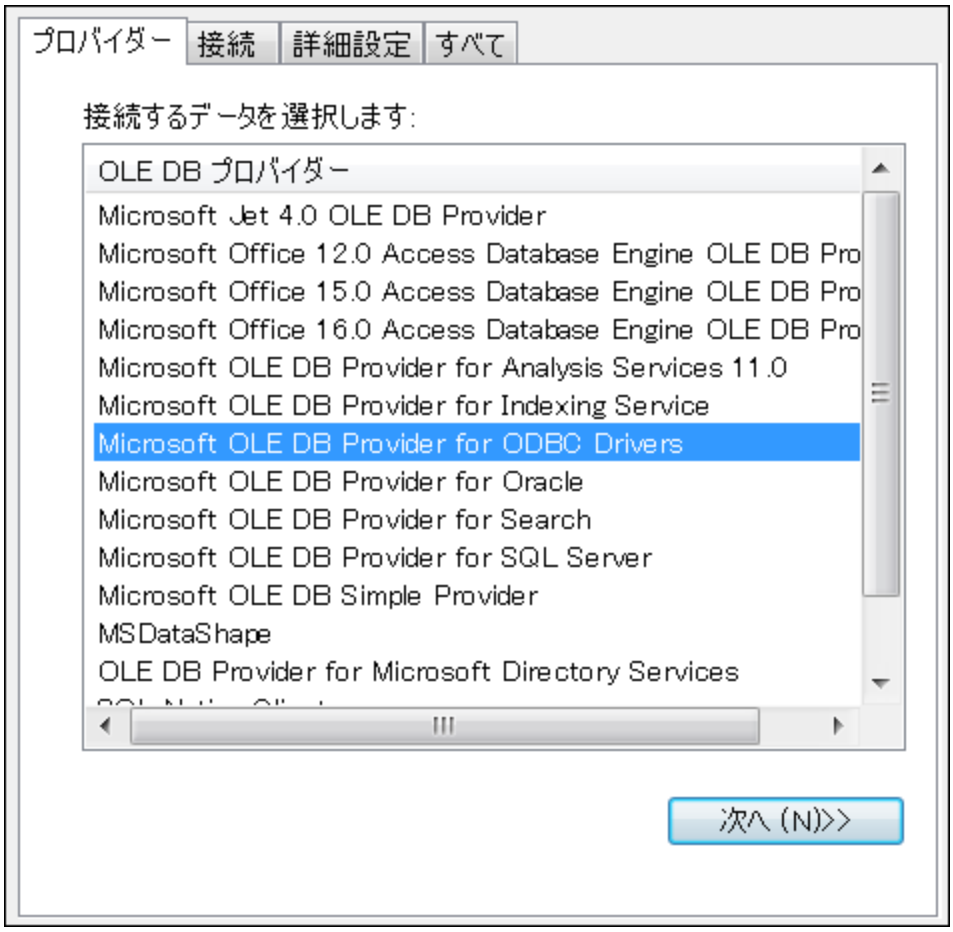
16.1.10.9 Microsoft Access (ADO) への接続

Microsoft Access データベースに接続するには、既存の Microsoft Access データベースに接続する[581]で説明されているとおり、データベースファイルのためのウィザードとブラウザーの手順を踏みます。別の方法としては、このトピックで説明されているとおり、明確的に ADO 接続をセットアップすることです。このアプローチは、データベースがパスワードで保護されている場合、役に立ちます。

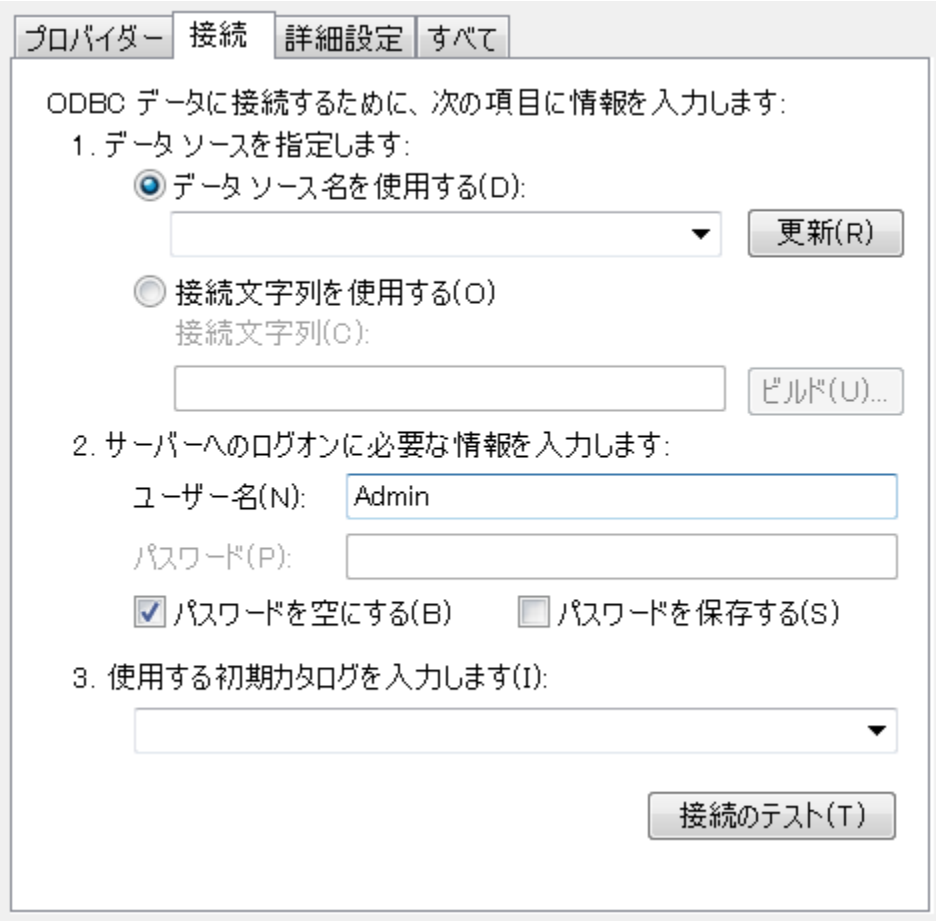
ODBC 接続を使用して、Microsoft Access に接続することもできますが、このシナリオには限度があるため、奨励されません。

パスワードで保護されている Microsoft Access データベースに接続する:

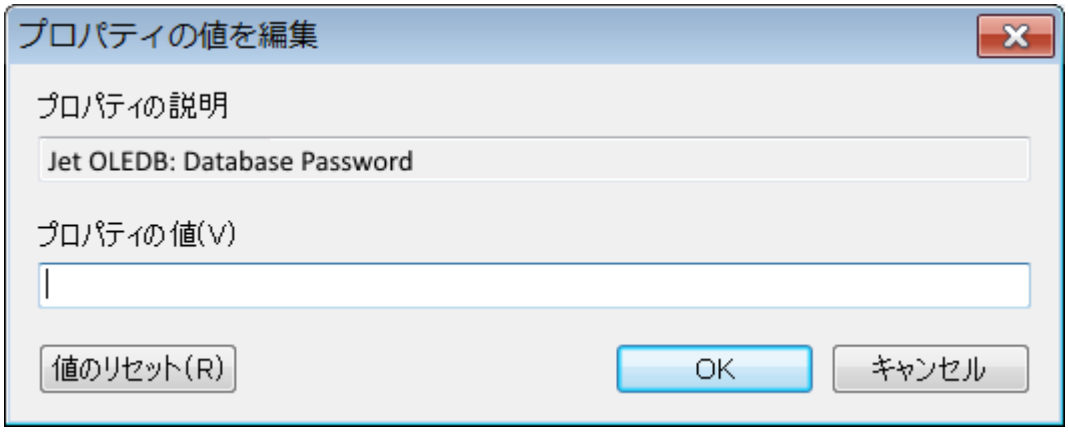
1. データベース接続ウィザードを開始します[574].
2. [ADO 接続] をクリックします。
3. 構築] をクリックします。



4. **[Microsoft Office 15.0 Access Database Engine OLE DB Provider]** を選択して、「次へ」をクリックします。



5. データソースボックス内でUNC フォーマットのMicrosoft Access ファイルへのパスを入力します。例えば、**myserver** がサーバーの名前であり **mynetworkshare** が共有するネットワークの名前である箇所での **\\myserver\\mynetworkshare\Reports\Revenue.accdb**。
6. 全て タブの **Jet OLEDB:Database Password** パスワード プロパティをダブルクリックし、データベースパスワードをプロパティの値として入力します。



メモ また接続できない場合、ワーキンググループ情報ファイル(**System.MDW**) を検索して、ユーザープロフィールに適用できるようにします。**Jet OLEDB: System database** プロパティの値を **System.MDW** ファイルのパスに設定します。

16.1.10.10 Microsoft SQL Server (ADO) への接続

このサンプルでは、ADO を使用したSQL Server データベースへの接続方法について説明されています。Https://docs.microsoft.com/en-us/sql/connect/oledb/download-oledb-driver-for-sql-server?view=sql-server-ver15 でダウンロード可能な奨励される **Microsoft OLE DB Driver for SQL Server (MSOLEDBSQL)**を使用する場合に適用される説明です。

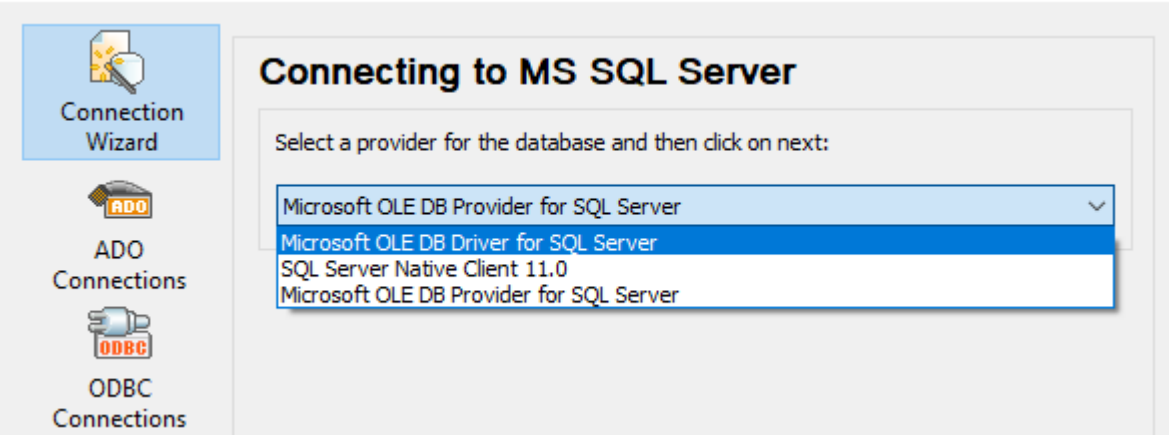
これらの説明を読む前に、ワークステーションに上記のプロバイダーがダウンロードされインストールされていることを確認してください。ADO プロバイダーはXMLSpy (32-ビット または64-ビット)のプラットフォームに一致する必要があることご注意してください。

SQL Server Native Client (SQLNCLI) または **Microsoft OLE DB Provider for SQL Server (SQLOLEDB)** などの他のADO プロバイダーを使用する場合、手順は同様ですが、これらのプロバイダーは奨励されていません。奨励されていないプロバイダーへの接続を成功させるには、SQL Server データリンクプロパティのセットアップ 582 で説明されている追加接続プロパティを設定する必要があります。

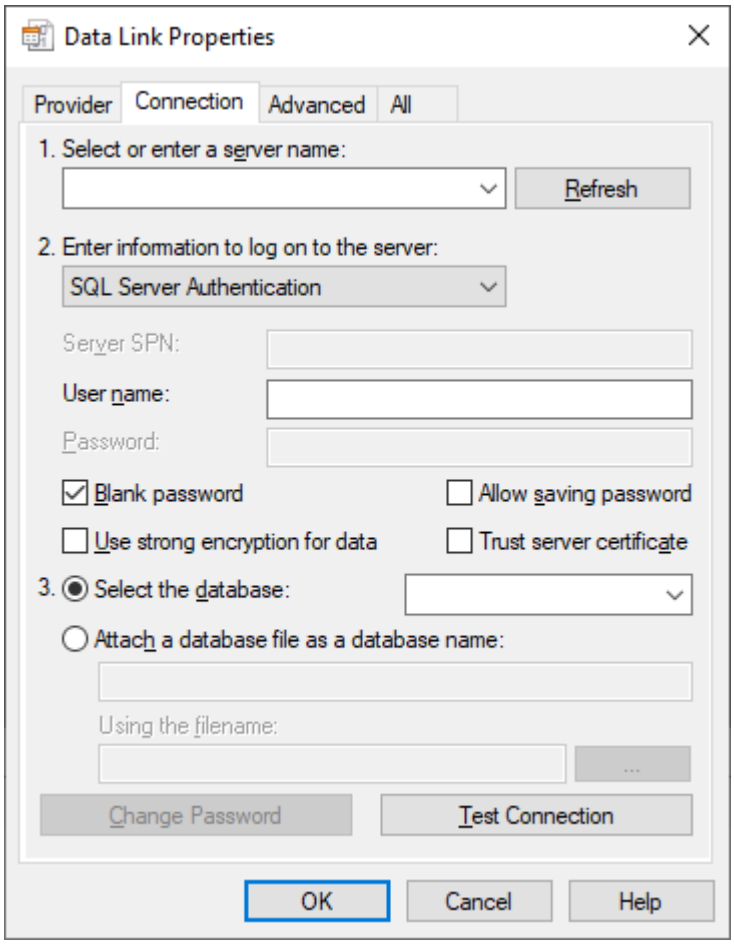
Microsoft OLE DB Provider for SQL Server (SQLOLEDB) は共通テーブル式 (CTE) とネストされたSELECT ステートメントなどの複雑なクエリのパラメーターバインディングに関する問題があることが知られています。

SQL サーバーに接続する方法:

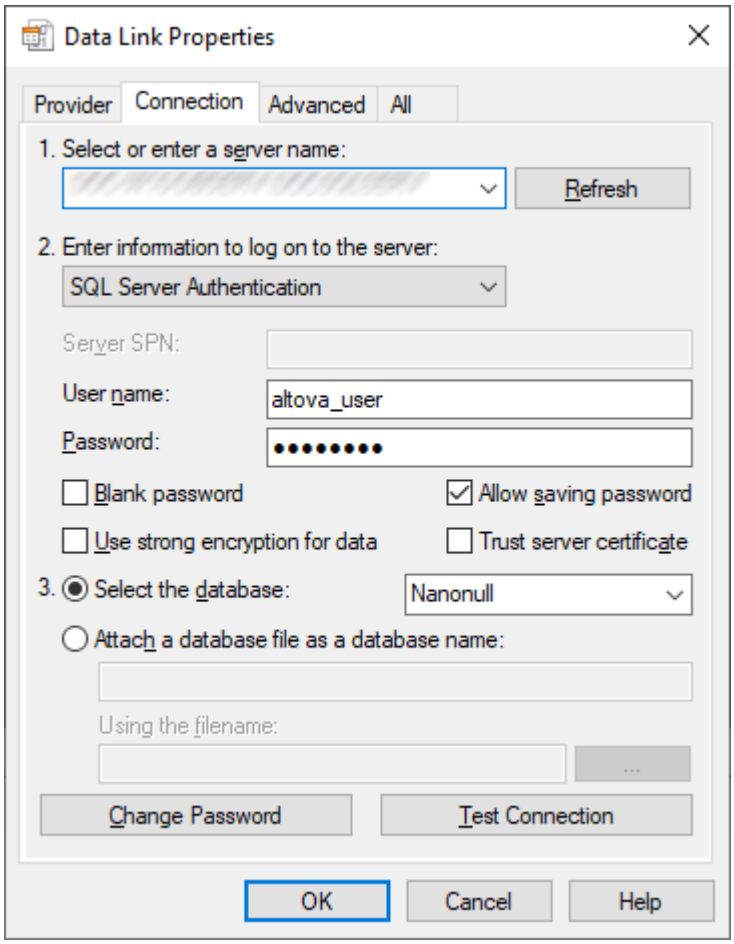
1. データベース接続ウィザードを開始します 574 。
2. **[Microsoft SQL Server (ADO)]** を選択して、[次へ] をクリックします。使用することのできるADO プロバイダーのリストが表示されます。このサンプルでは **Microsoft OLE DB Driver for SQL Server** が使用されています。リストに存在しない場合、上記の通りコンピューターに接続されていることを確認してください。



3. 「次へ」をクリックします。データリンクプロパティダイアログボックスが表示されます。



4. データベースサーバーの名前を選択、または入力します。例えば、**SQLSERV01**。名前を持つSQL Server インスタンスに接続する場合、サーバー名は**SQLSERV01\SOMEINSTANCE** になります。
5. データベースサーバーがWindows ドメイン上の認証されているユーザーの接続を許可している場合、「**Windows 認証**」を選択してください。それ以外の場合は、[**SQL Server 認証**]を選択し、[**空のパスワード**] チェックボックスをクリアし、対応するボックスにデータベース証明書を入力します。
6. **パスワードの保存の許可** チェックボックスを選択し、接続するデータベースを選択します（このサンプル内では、Nanonull です）。



7. この時点で接続をテストする場合は、[**接続のテスト**]をクリックします。これは任意の奨励されるステップです。
8. **[OK]**をクリックします。

16.1.10.11 Microsoft SQL Server (ODBC) への接続

このサンプルでは、ODBC を使用したSQL Server データベースへの接続方法について説明されています。

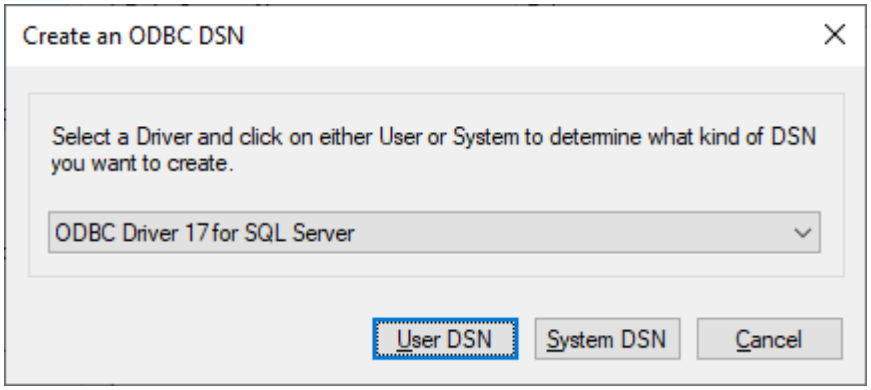
必要条件:

- **Microsoft ODBC Driver for SQL Server** をMicrosoft W eb サイトからダウンロードしてインストールします。https://docs.microsoft.com/en-us/SQL/connect/odbc/download-odbc-driver-for-sql-server を参照してください。このサンプルは**Microsoft ODBC Driver 17 for SQL Server** を**SQL Server 2016** データベースに接続するために使用しています。接続するSQL Server のバージョンにより異なるODBC ドライバーバージョンをダウンロードする必要がある場合があります。使用中のSQL Server によりサポートされるODBC ドライバーバージョンに関する情報は、システムの必要条件を参照してください。

ODBC を使用してのSQL Server への接続:

1. データベース接続ウィザードを開始します[574]。

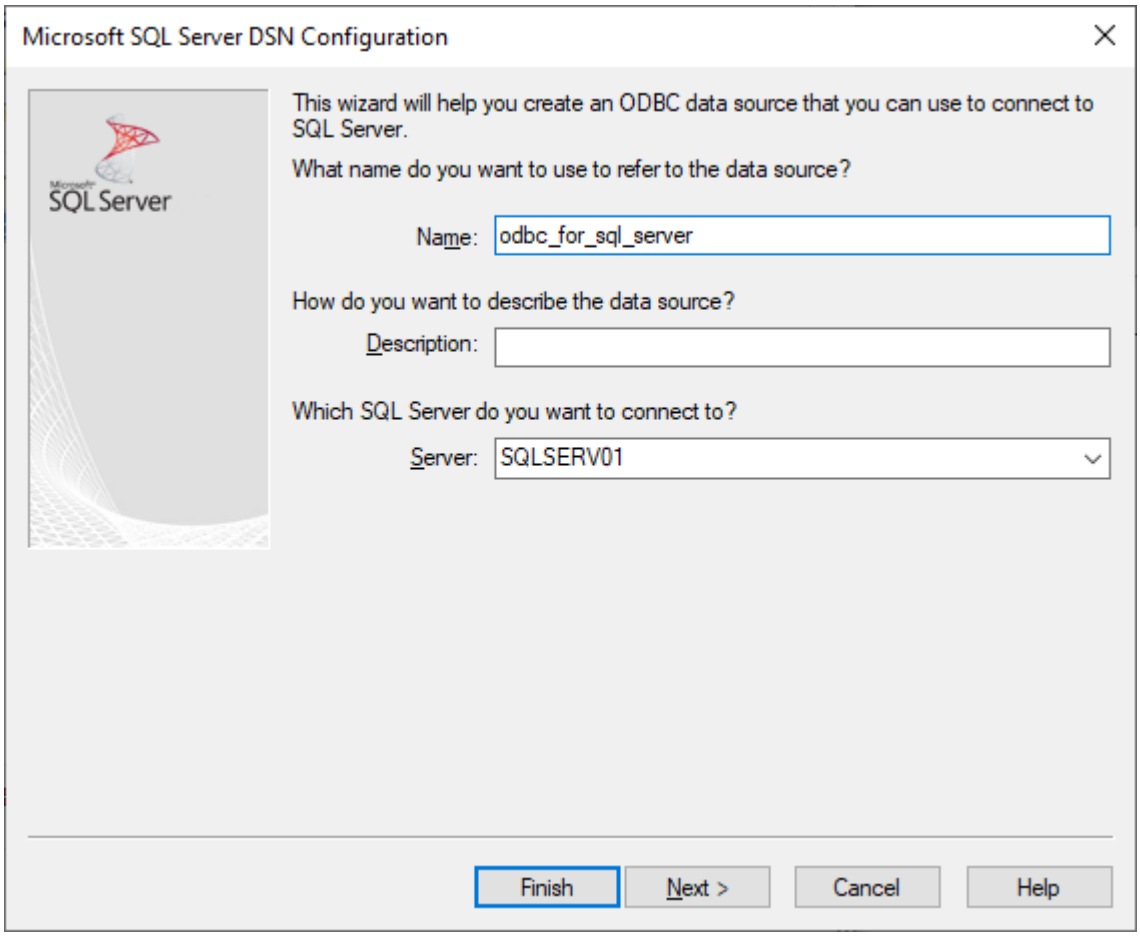
2. [ODBC 接続] をクリックします。
3. [ユーザーDSN] を選択して、(または、管理者の特権がある場合、 [システムDSN] を選択して、)[追加] をクリックします。
4. ドライバーをリストから選択します。インストール後にのみドライバーはリスト内に表示されることご注意してください。



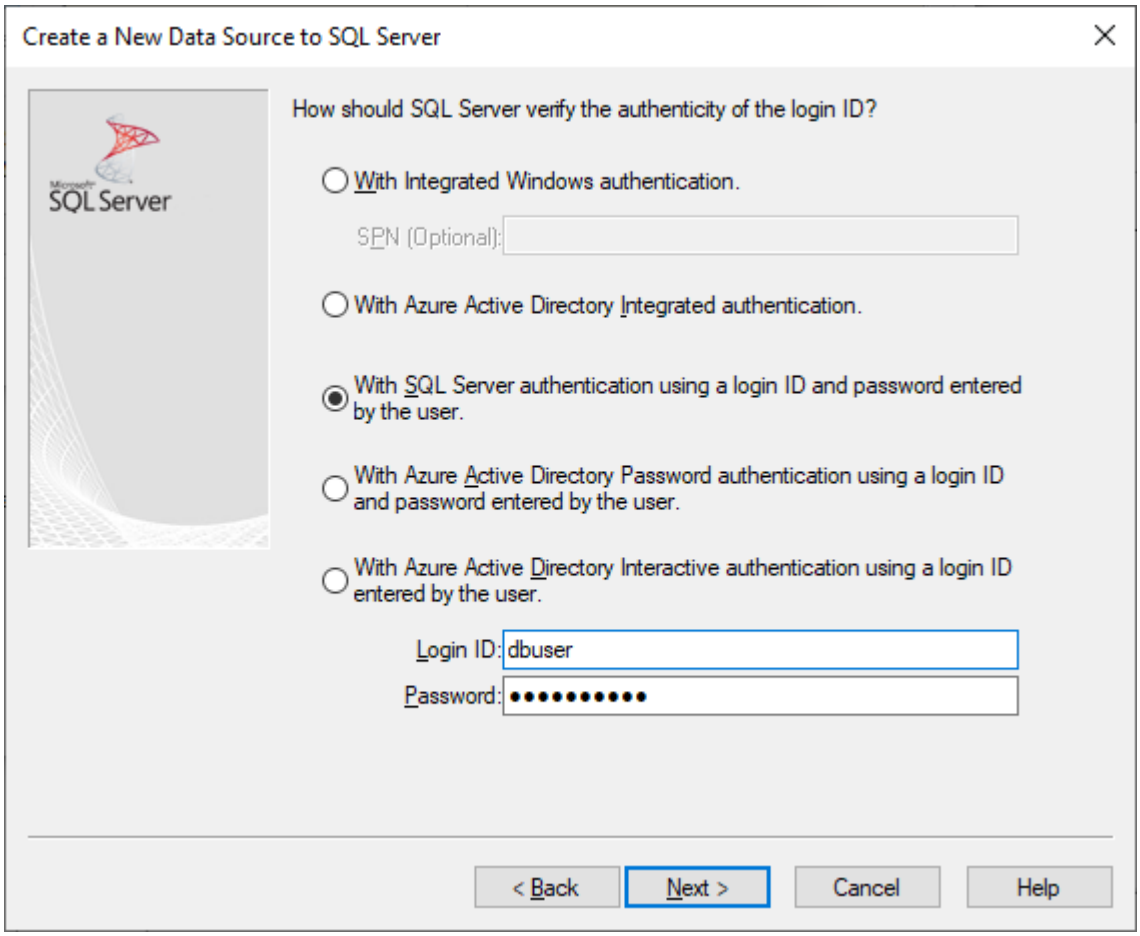
5. [ユーザーDSN] (またはシステムDSN を作成する場合) [システムDSN]をクリックします。

> XMLSpy が管理者として作動することを必要とするシステムDSN の作成方法このため**System DSN** を作成するには、ウィザードをキャンセルし、XMLSpy が管理者として作動していることを確認し、上記のステップを再度実行します。

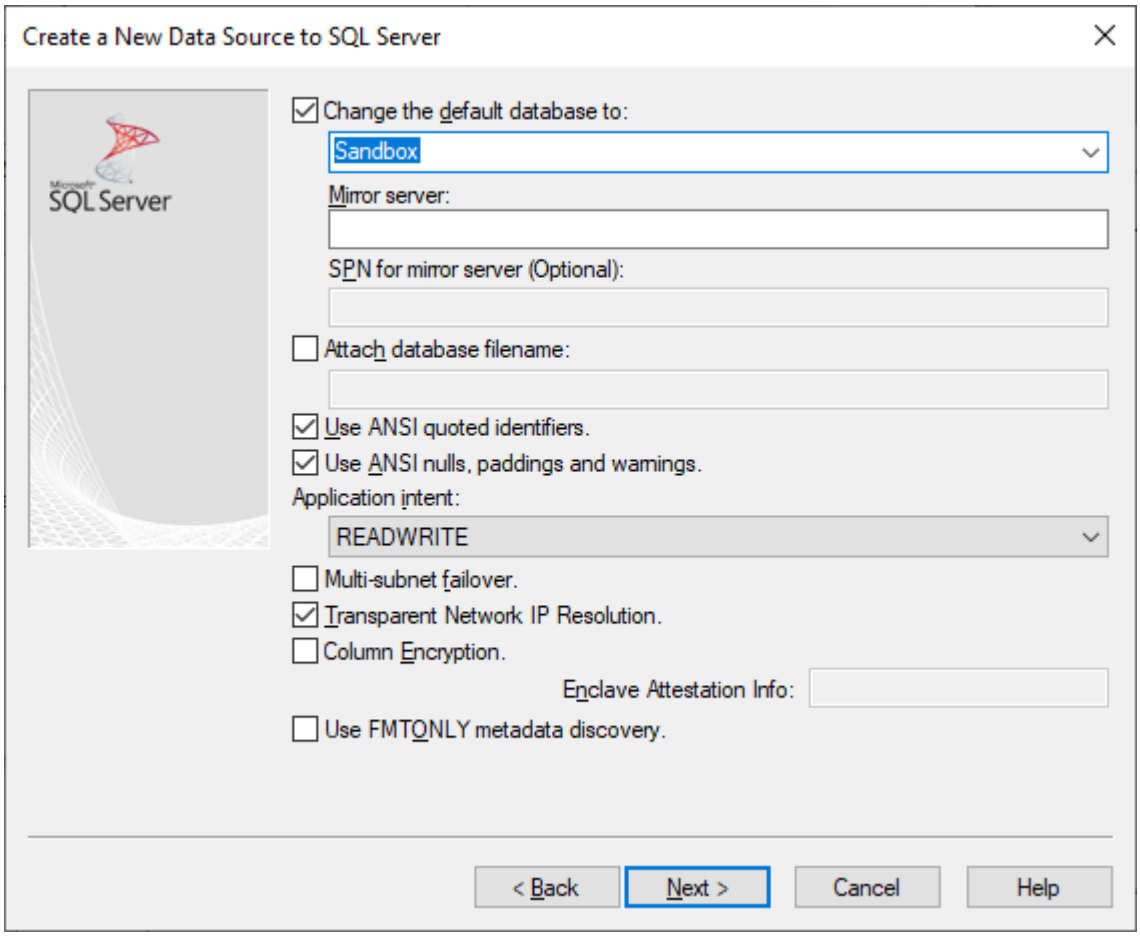
6. 名前を入力して、任意でこの接続のDの説明を入力し、接続するSQL Server (この場合 **SQLSERV01**)をリストから選択します。.



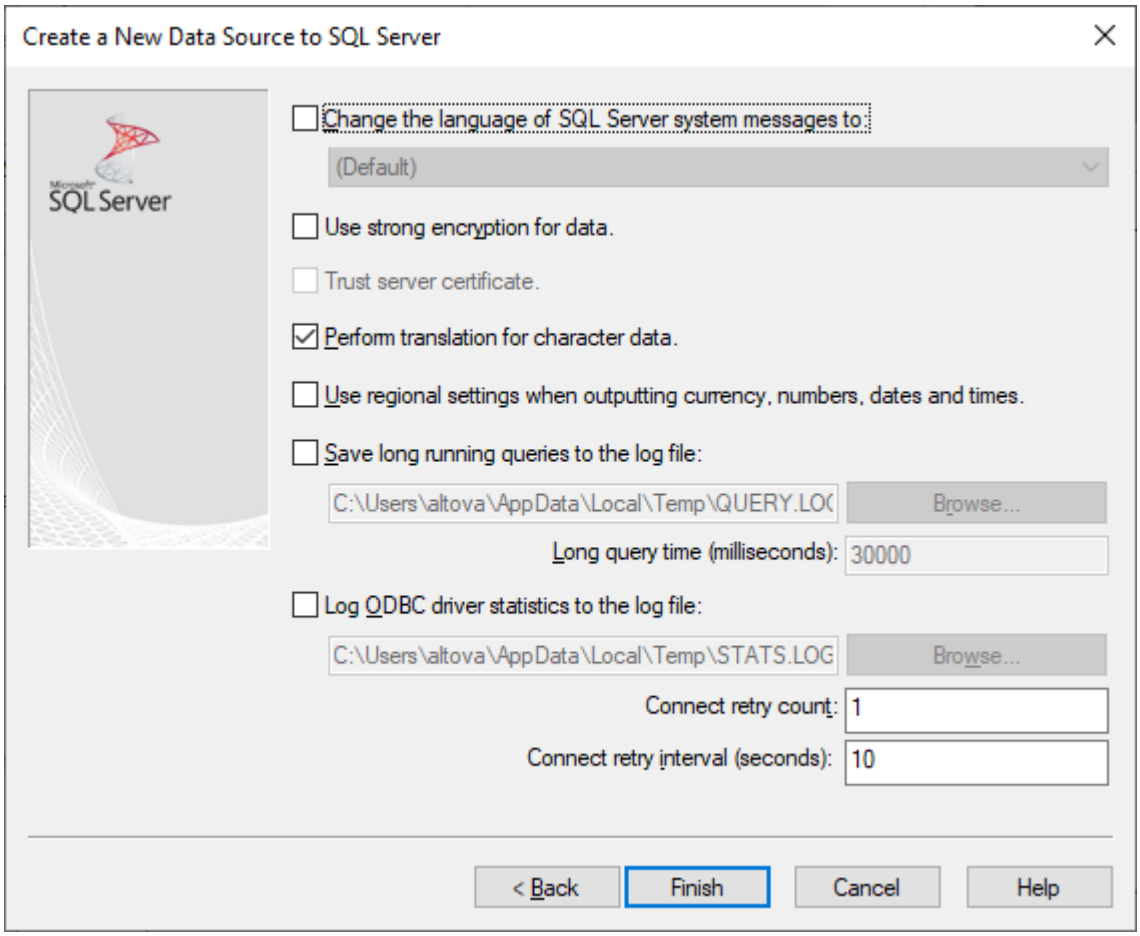
7. データベースサーバーがWindowsドメイン上の認証されているユーザーの接続を許可している場合、「**統合されたWindows認証**」を選択してください。それ以外の場合、適用可能なオプションから1つを選択してください。このサンプルは **SQL Server認証**...を使用しています。これはユーザー名とパスワードが関連するボックスに入力されることを必要としています。



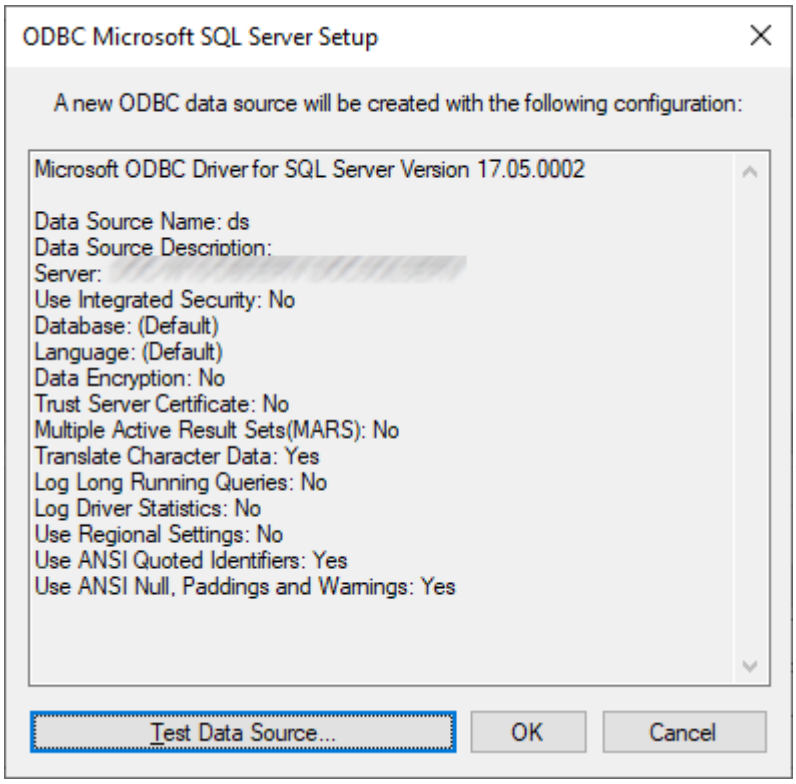
8. オプションで **デフォルトのデータベースを変更** チェックボックスを選択して、接続するデータベースの名前を入力します（このサンプルでは**Sandbox**）。



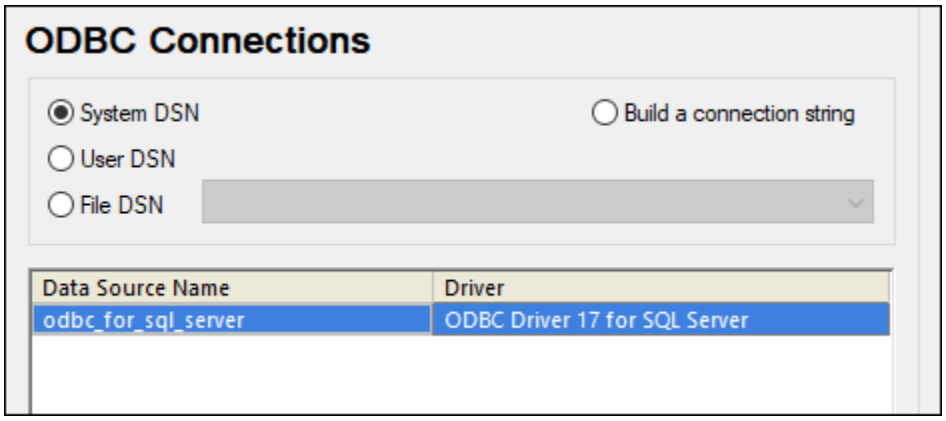
9. ［次へ］をクリックして、任意で、この接続のための追加パラメーターを構成します。



10. 「完了」をクリックします。接続の詳細をリストした確認ダイアログボックスが開かれます。



11. **[OK]**をクリックします。構成された通り ユーザー または システム データソースのリスト内でデータソースが表示されます。例:



16.1.10.12 MySQL (ODBC) への接続

このサンプルではODBC ドライバーを使用して、Windows マシンから MySQL データベースへ接続する方法について説明されています。Windows からは、MySQL ODBC ドライバーを入手することができませんので、個別にダウンロードおよびインストールする必要があります。このサンプルでは、MySQL コネクタ/ODBC 8.0 を使用しています。

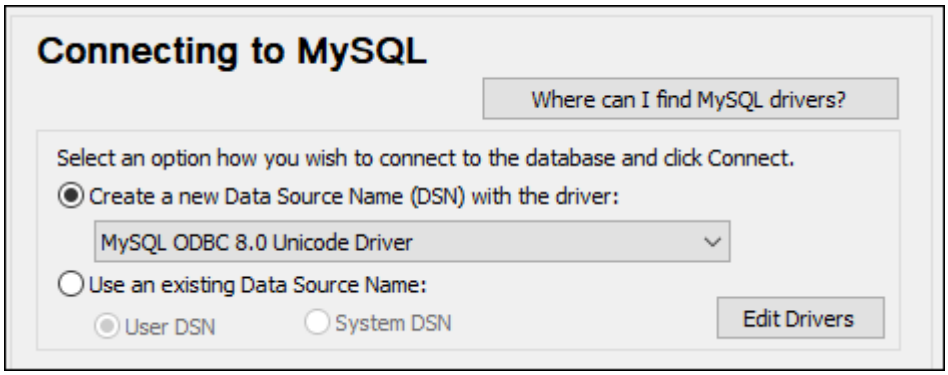
前提条件:

- MySQL ODBC ドライバーをオペレーティングシステム上にインストールする必要があります。使用するデータベースサーバーバージョンのために奨励されるドライバーバージョンのためのMySQLドキュメントを確認してください (https://dev.mysql.com/downloads/connector/odbc/ を確認してください)。
- データベース接続の詳細: ホスト、データベース、ユーザー名およびパスワード

64-ビットプラットフォームのためにMySQL Connector/ODBC をインストールしている場合、64-ビットプラットフォームのためにXMLSpy をインストールしてください。

ODBC を介して MySQL に接続する方法:

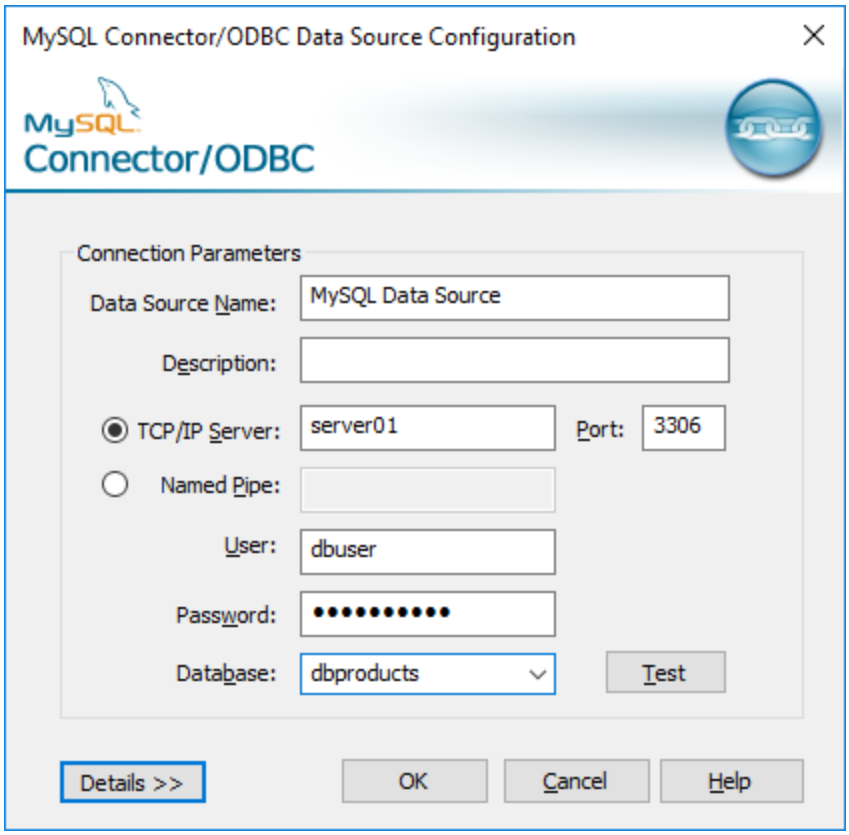
1. データベース接続ウィザードを開始します 574。
2. **MySQL (ODBC)** を選択して、「次へ」をクリックします。



3. [ドライバーを使用して新しいデータソース名 (DSN) を作成] を選択して、MySQLドライバーを選択します。MySQLドライバーがリストにない場合は、[ドライバーを編集] をクリックして、使用可能なMySQLドライバーを選択します (使用中のオペレーティングシステムにインストールされているすべてのODBC ドライバーがリストされています)。

XMLSpy 64-ビットをインストールすると、64-ビット ODBC ドライバーがリスト内に表示されます。それ以外の場合、32-ビット ODBC ドライバーが表示されます。使用可能な ODBC ドライバーの確認 593 も参照してください。

4. 「接続」をクリックします。



5. データソース名ボックスの中に、今後このODBC データソースを認識するために役に立つ詳細を入力します。
6. データベース接続資格情報（TCP/IP サーバー、ユーザー、パスワード）を入力して、データベースを選択し、「OK」をクリックします。

メモ　データベースサーバーがリモートの場合、サーバー管理者が使用中のコンピューターのIP アドレスからのリモート接続を承諾するように構成される必要があります。また、[詳細>>] をクリックすると使用することのできる追加パラメーターが表示されます。デフォルトの値を変更する前に、ドライバーのドキュメントを確認してください。

16.1.10.13 Oracle (JDBC) への接続

JDBC インターフェイスを使用して、クライアントマシンから、Oracle データベースサーバーへの接続の方法を説明します。接続は、Oracle Web サイトから使用することのできる、**Oracle Instant Client Package (Basic)** を使用する、Java 接続として作成されます。接続の型の利点は、Oracle Instant Client Package により与えられる Java 環境と .jar ライブラリのみを必要とし、更に複雑なデータベースクライアントをインストールする手間を省くことができる必点です。

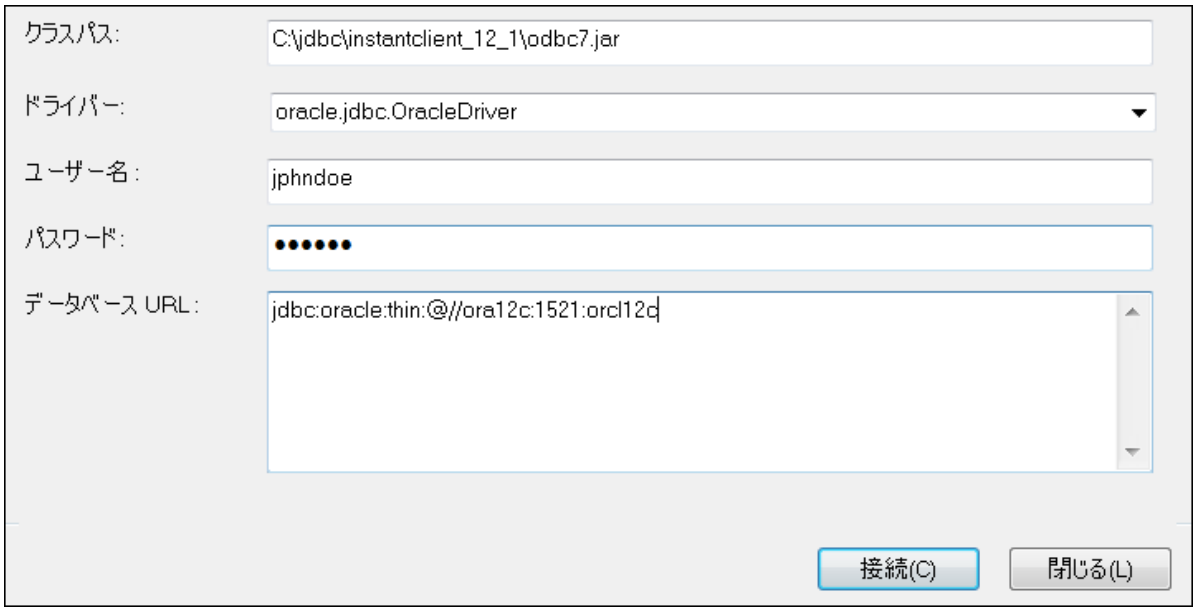
前提条件:

- JRE (Java Runtime Environment) または Java Development Kit (JDK) がインストールされている必要があります。これは Oracle JDK または Oracle OpenJDK などのオープンソースビルドであることができます。XMLSpy は次のロケーションから Java Virtual Machine (JVM) へのパスを次の順序で決定します: a) アプリケーションオプション内で設定したカスタム JVM パス b) Windows レジストリ内で見つかった JVM パス。c) `JAVA_HOME` 環境変数。
- XMLSpy (32-ビット、64-ビット) のプラットフォームが JRE/JDK のプラットフォームに一致することを確認してください。
- **Oracle Instant Client Package (Basic)** が使用中のオペレーティングシステムで使用可能である必要があります。パッケージは、Oracle の公式 Web サイトからダウンロードすることができます。このサンプルは、Windows 32-ビットのための Oracle Instant Client Package バージョン 12.1.0.2.0 、そして結果的に Oracle JDK 32-ビットを使用します。

- データベース接続の詳細があります: ホスト、データベースパス、またはエイリアス、ユーザー名、およびパスワード。

クライアントパッケージを使用して Oracle に接続する:

1. データベース接続ウィザードを開始します[574]。
2. 「JDBC 接続」をクリックします。
3. 「クラスパス」の横にデータベースへの接続を与える .jar ファイルへのパスを入力します。必要であれば、セミコロンで区切られた .jar ファイルパスのリストを入力することもできます。このサンプルでは、必要とされる .jar ファイルは次のパスを使用して見つけることができます: **C:\jdbc\instantclient_12_1\ojdbc7.jar**。.jar ファイルパスをオペレーティングシステムの CLASSPATH 環境変数に追加した場合、「クラスパス」テキストボックスを空白のままにすることができます。(次も参照: CLASSPATH の構成[596])。
4. 「ドライバー」ボックスから、**oracle.jdbc.OracleDriver** または **oracle.jdbc.driver.OracleDriver** を選択します。このエントリは、有効な .jar ファイルパスが、「クラスパス」テキストボックス内、または、オペレーティングシステムの CLASSPATH 環境変数内で検出された場合のみ使用することができます(前のステップを参照してください)。
5. データベースへのユーザー名とパスワードを対応するテキストボックスに入力します。



6. ハイライトされた値を、使用するデータベースサーバーに適用する値と置き換えて、データベース URL テキストボックス内にデータベースサーバーへの接続文字列を入力します。

```
jdbc:oracle:thin:@//host:port:service
```

7. 「接続」をクリックします。

16.1.10.14 Oracle (ODBC) への接続

このサンプルでは、ローカルオペレーティングシステム上にインストールされている Oracle データベースクライアントを介して、ネットワークマシン上で XMLSpy から Oracle データベースサーバーへの接続のよく使用されている状況について説明されています。

このサンプルでは、XMLSpy 内のデータベース接続ウィザードを使用して、ODBC データソース(DSN) のセットアップについて説明されています。既に DSN が作成されている場合は Windows 内の **ODBC データソース管理者**から直接作成しプロンプトされたら選択する事も可能です。ODBC データソースの詳細に関しては、ODBC 接続のセットアップ[591]を参照してください。

必要条件:

- (ODBC Oracle ドライバーを含む) Oracle データベースクライアントが使用中のオペレーティングシステムでインストール、および構成されていなければなりません。Oracle データベースクライアントのインストールおよび構成方法は Oracle ソフトウェアに付随するドキュメントを参照してください。
- **tnsnames.ora** ファイルのパスは Oracle ホームディレクトリがインストールされている場所によります。Oracle データベースクライアント 11.2.0 に関しては、デフォルト Oracle ホームディレクトリのパスは以下の通りです:

```
ORCL =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP)(HOST = server01)(PORT = 1521))
    )
    (CONNECT_DATA =
      (SID = orcl)
      (SERVER = DEDICATED)
    )
  )
```

tnsnames.ora ファイルのパスは Oracle ホームディレクトリがインストールされている場所によります。Oracle データベースクライアント 11.2.0 に関しては、デフォルト Oracle ホームディレクトリのパスは以下の通りです:

```
C:\app\username\product\11.2.0\client_1\network\admin\tnsnames.ora
```

接続の詳細を貼り付け保存する、または使用可能であれば Oracle *ネット構成アシスタント*ウィザード を実行することにより **tnsnames.ora** ファイルに新規エントリを追加することができます。

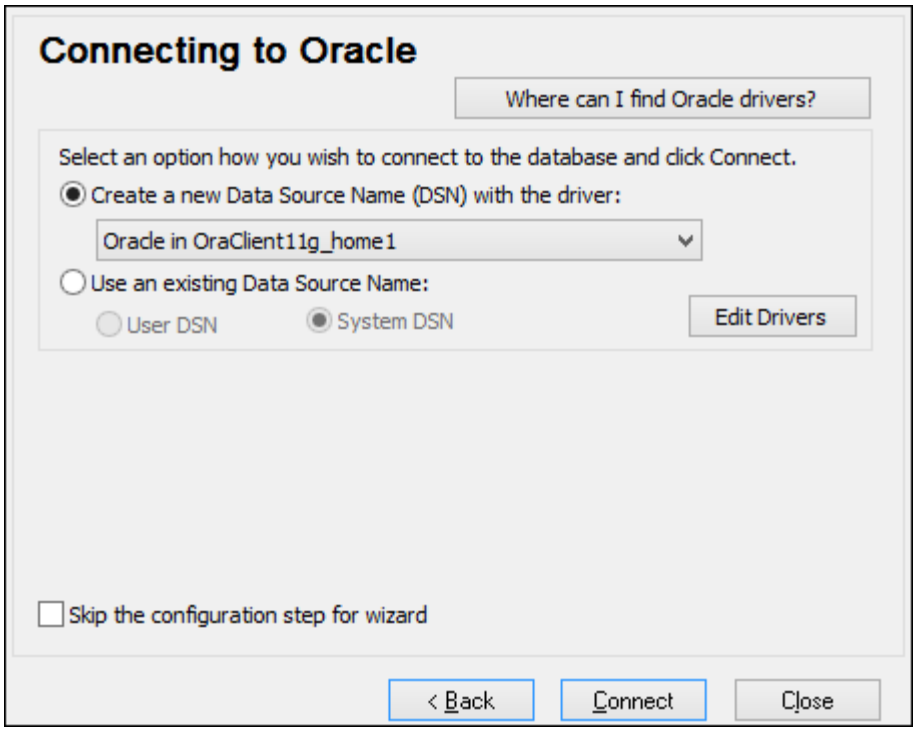
ODBC を使用しての Oracle への接続:

1. データベース接続ウィザードを開始します 574.
2. **[Oracle (ODBC / JDBC)]** を選択して、「次へ」をクリックします。

3. **[ODBC]** を選択します。

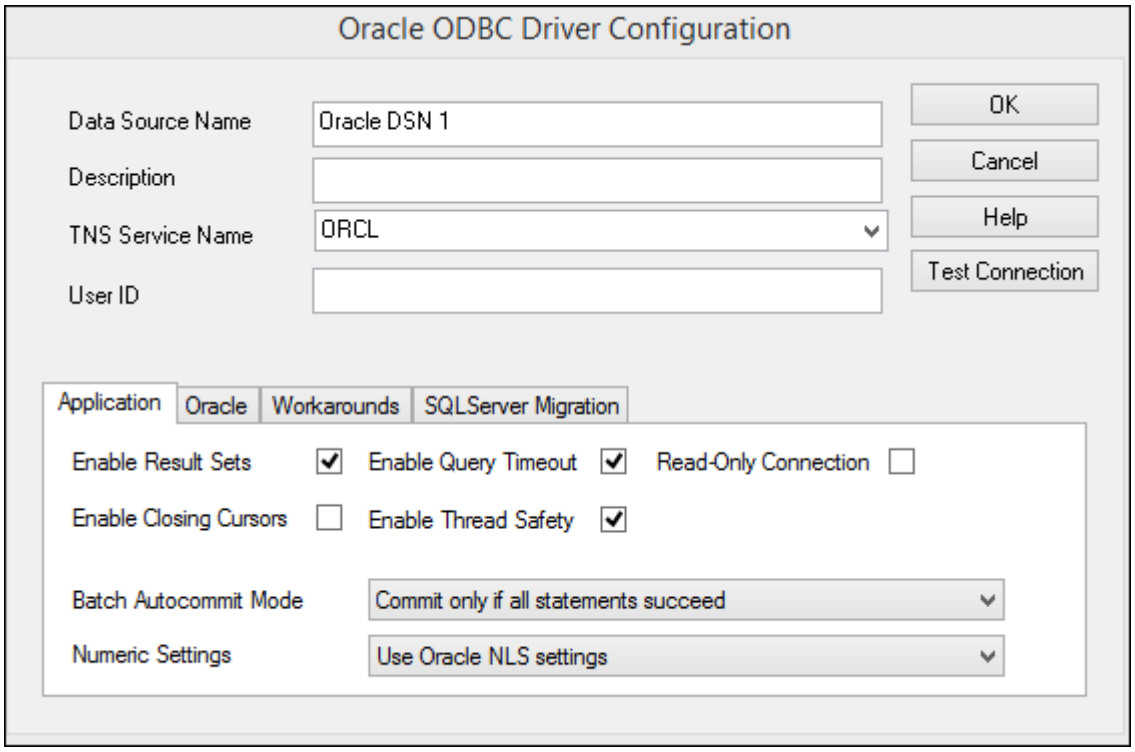
4. [ドライバーの編集] をクリックします。

5. 使用するOracleドライバーを選択します（このサンプルでは**Oracle in OraClient11g_home1**です）。Oracle クライアントをインストール後システムで使用可能なOracleドライバーのリストを表示します。
6. [戻る] をクリックします。
7. [新しいデータソース（DSN）をドライバーを使用して作成] を選択して、ステップ4 で選択されたOracleドライバーを選択します。

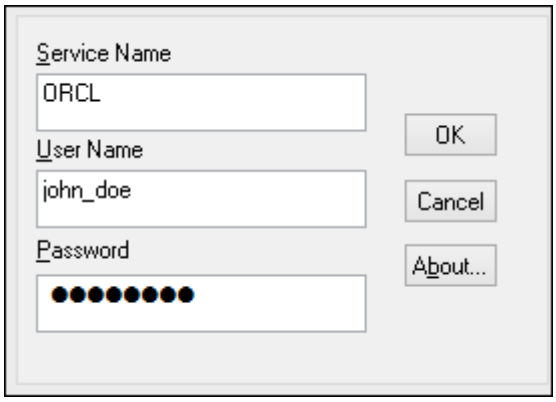


Microsoft ODBC for Oracle という名称のMicrosoft により提供されるドライバーを使用することは避けてください。Microsoft はOracle により提供されるODBC ドライバーを使用することを奨励しています。(以下参照: http://msdn.microsoft.com/en-us/library/ms714756%28v=vs.85%29.aspx)。

8. 「接続」をクリックします。



9. データソース名テキストボックスにデータソースを識別する名前を入力します（このサンプルの場合は**Oracle DSN 1** です）。
10. TNS サービス名ボックスでは、**tnsnames.ora** ファイルで定義されている通りの接続名を入力します。（prerequisites 639 前提条件を参照してください）。このサンプルでは接続名 **ORCL** です。
11. 「OK」をクリックします。



12. データベースのユーザー名とパスワードを入力して、「OK」。をクリックします。

16.1.10.15 PostgreSQL (ODBC) への接続

このトピックではODBC ドライバーを使用した、Windows マシンから PostgreSQL データベースへの接続のサンプル手順について説明されています。Windows からは、 PostgreSQL ODBC ドライバーを入手することができませんので、個別にダウンロードおよびインストールする必要があります。このサンプルでは、公式 Web サイトからダウンロードされたpsqlODBC ドライバー（バージョン 11.0）を使用しています。（データベースドライバー概要 575 も参照してください）。

メモ　（ODBC ドライバー無しで直接 PostgreSQL データベースサーバーに接続するには、次を参照してください：PostgreSQL 接続のセットアップ[598]。

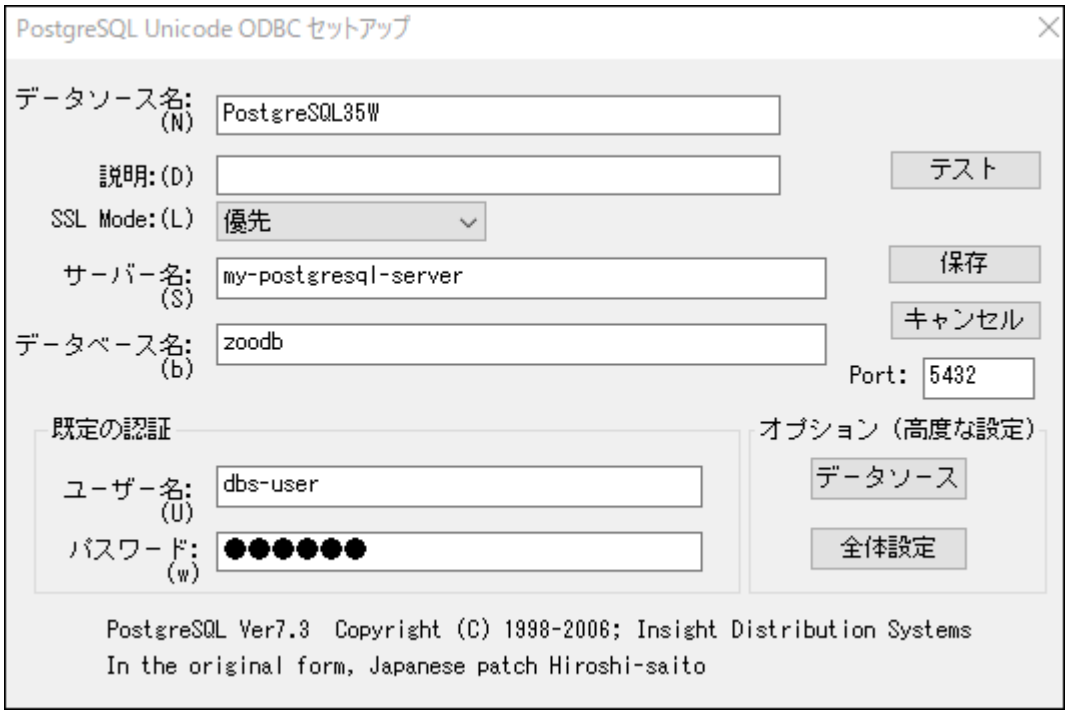
前提条件:

- *psqlODBC* ドライバーを使用中のオペレーティングシステムにインストールする必要があります。
- データベース接続の詳細：ホスト、データベース、ユーザー名およびパスワード

ODBC を使用した PostgreSQL への接続をセットアップする方法:

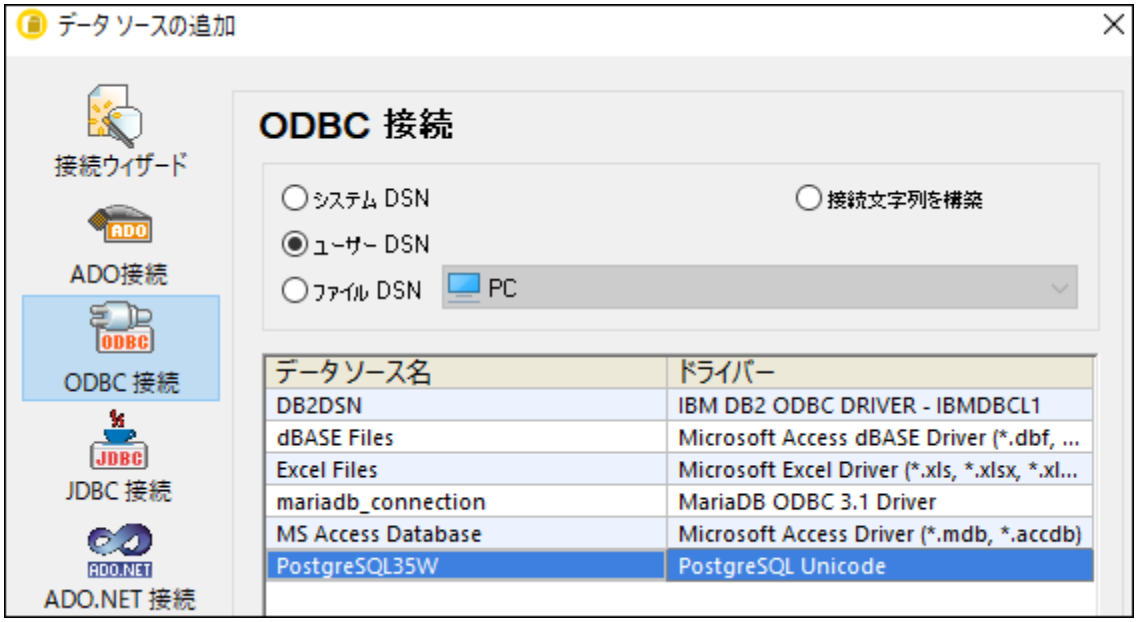
1. データベース接続ウィザードを開始します。[574]
2. **PostgreSQL (ODBC)** をクリックします。
3. ユーザー DSN オプションを選択します。
4. 「新規の DSN の作成」 をクリックし、ドロップダウンリストからドライバーを選択します。PostgreSQL ドライバーがリスト内で使用できない場合、上記の必要条件で説明されているとおり、PostgreSQL ODBC ドライバーがオペレーティングシステム上にインストールされていることを確認してください。

5. 「ユーザー DSN」をクリックします。



6. データベース資格情報を入力し、「保存」をクリックします（これらはデータベースオーナーにより提供されています）。

接続はODBC 接続のリスト内で使用できるようになります。データベースに接続するには、接続をダブルクリックし、「接続」をクリックします。



16.1.10.16 Progress OpenEdge (JDBC) への接続

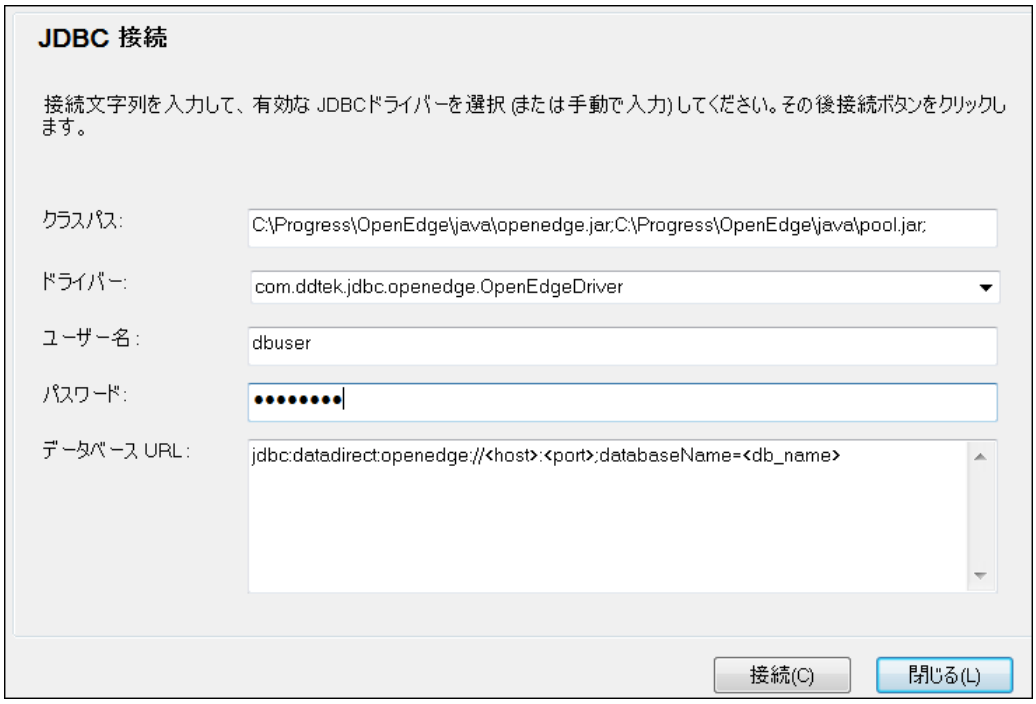
JDBC を使用して Progress OpenEdge 11.6 データベースサーバーへの接続するためのサンプルの手順が説明されています。

前提条件:

- JRE (Java Runtime Environment) または Java Development Kit (JDK) がインストールされている必要があります。これは Oracle JDK または Oracle OpenJDK などのオープンソースビルドであることができます。XMLSpy は次のロケーションから Java Virtual Machine (JVM) へのパスを次の順序で決定します: a) アプリケーション オプション内で設定したカスタム JVM パス b) Windows レジストリ内で見つかった JVM パス。c) `JAVA_HOME` 環境変数。
- XMLSpy (32-ビット、64-ビット) のプラットフォームが JRE/JDK のプラットフォームに一致することを確認してください。
- オペレーティングシステムの `PATH` 環境変数は、JRE または JDK インストールの `bin` ディレクトリへのパスを含んでいる必要があります。例、`C:\Program Files (x86)\Java\jre1.8.0_51\bin`.
- Progress OpenEdge JDBC ドライバーが使用中のオペレーティングシステムで使用可能である必要があります。このサンプルでは、JDBC 接続は、**C:\Progress\OpenEdge\java** に OpenEdge SDK のインストールの一部として存在する **openedge.jar** と **pool.jar** ドライバーコンポーネントファイルにより与えられています。.
- データベース接続の詳細があります: ホスト、データベース パス、またはエイリアス、ユーザー名、およびパスワード。

JDBC を使用して OpenEdge へ接続する:

1. データベース接続ウィザードを開始します574.
2. 「**JDBC 接続**」をクリックします。
3. 「クラス パス」の横にデータベースへの接続を与える .jar ファイルへのパスを入力します。必要であれば、セミコロンで区切られた .jar ファイル パスのリストを入力することもできます。このサンプルでは、必要とされる .jar ファイルは次のパスを使用して見つけることができます`C:\Progress\OpenEdge\java\openedge.jar;C:\Progress\OpenEdge\java\pool.jar;`。.jar ファイル パスをオペレーティングシステムの CLASSPATH 環境変数に追加した場合、「クラス パス」テキスト ボックスを空白のままにすることができます。(次も参照: CLASSPATH の構成596)。
4. 「ドライバー」ボックスから、**com.ddtek.jdbc.openedge.OpenEdgeDriver** を選択します。このエントリは、有効な .jar ファイル パスが、「クラス パス」テキスト ボックス内、または、オペレーティングシステムの CLASSPATH 環境変数内で検出された場合のみ使用することができます (前のステップを参照してください)。



5. データベースへのユーザー名とパスワードを対応するテキストボックスに入力します。
6. ハイライトされた値を、使用するデータベースサーバーに適用する値と置き換えて、データベースURL テキストボックス内にデータベースサーバーへの接続文字列を入力します。

```
jdbc:datadirect:openedge://host:port;databaseName=db_name
```

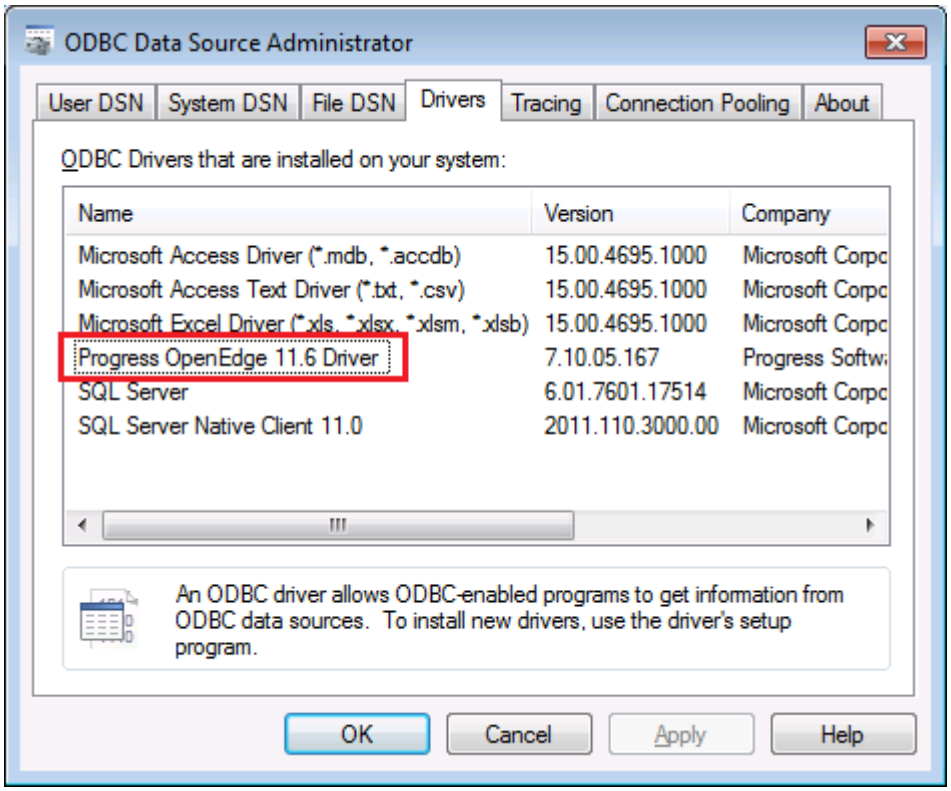
7. 「接続」をクリックします。

16.1.10.17 Progress OpenEdge (ODBC) への接続

このサンプルではProgress OpenEdge 11.6 ODBC ドライバーを使用して、Progress OpenEdge データベースサーバーへの接続の簡単な手順について説明されています。

前提条件:

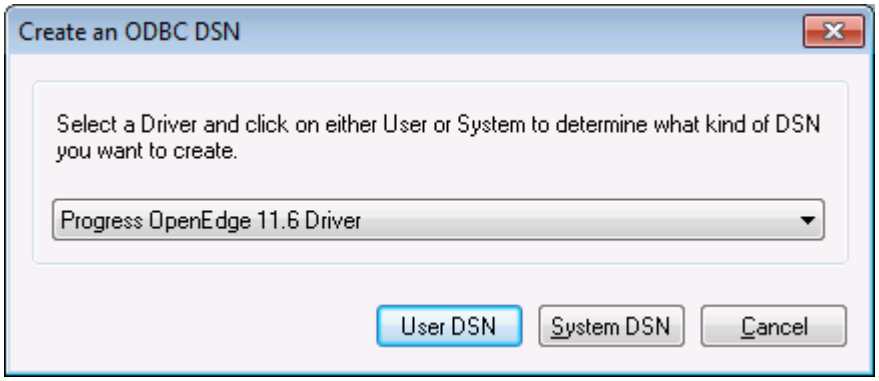
- *Progress OpenEdge ドライバーのための ODBC コネクタ*が使用中のオペレーティングシステムにインストールされている必要があります。Progress OpenEdge ODBC ドライバーは、ベンダーのW eb サイトからダウンロードすることができます(次も参照: データベースドライバーの概要 575)。XMLSpy の32 ビットバージョンを作動する場合は、32 ビットドライバーをダウンロードし、64 ビットバージョンを作動する場合は、64 ビットドライバーをダウンロードしてください。インストールの後、ODBC ドライバーが使用中のマシンで使用することができるかを確認してください(次も参照: 使用することのできるODBC ドライバーを確認する 593)。



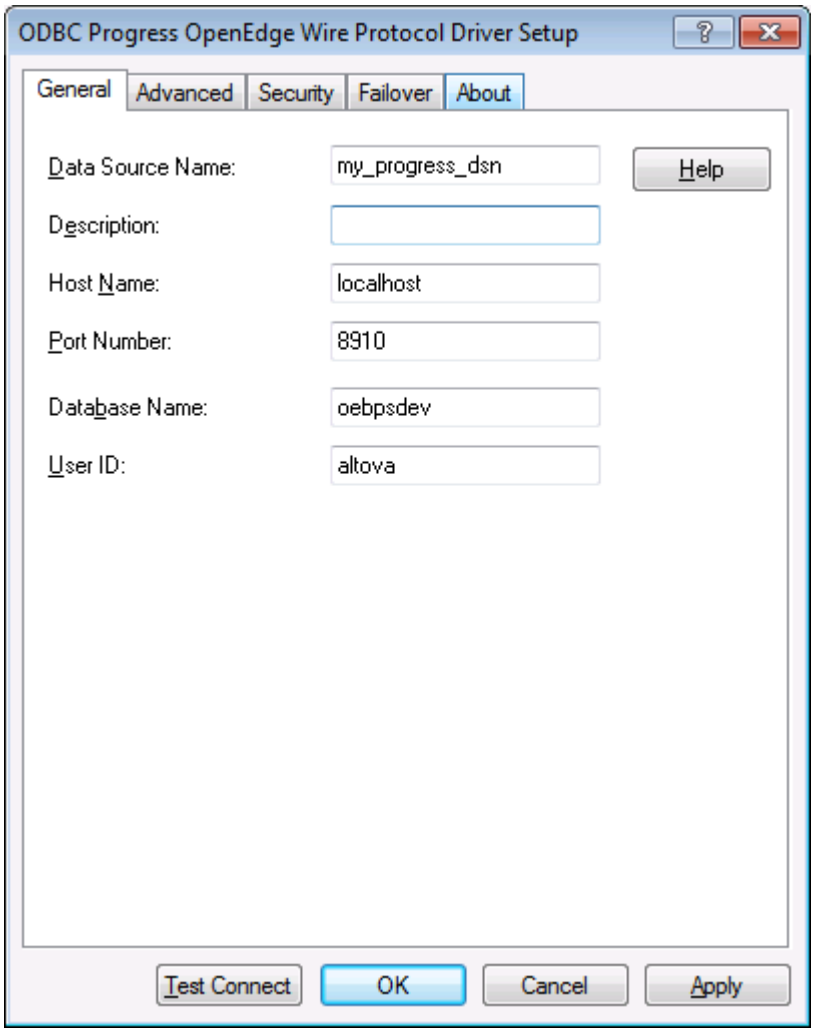
- データベース接続の詳細があります: ホスト名、ポート番号、データベース名、ユーザーID、およびパスワード。

ODBC を使用して Progress OpenEdge に接続する:

1. データベース接続ウィザードを開始します 574.
2. 「ODBC 接続」をクリックします。
3. 「ユーザーDSN」をクリックします。(または、次の手順が類似している「システムDSN」、または、「ファイルDSN」をクリックします)。
4. 「追加」 をクリックします。
5. 「Progress OpenEdge ドライバー」をリストから選択し、「ユーザーDSN」(または、適用できる場合は、「システムDSN」)をクリックします。

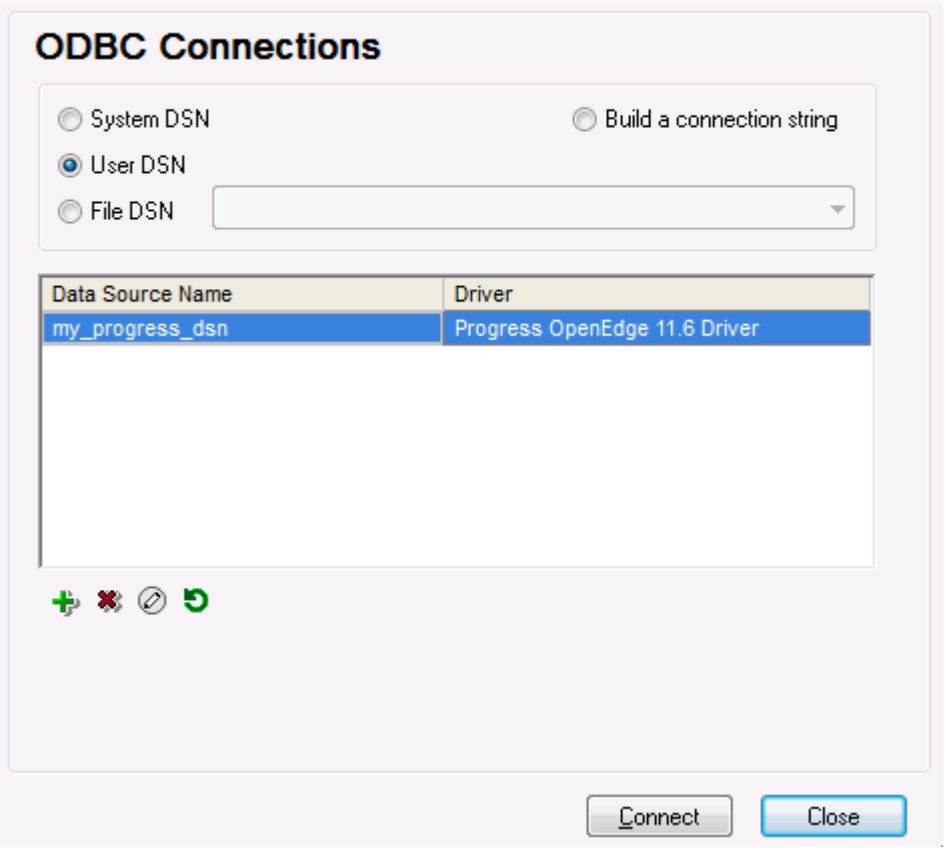


6. データベース接続の資格情報（データベース、サーバー、ポート、ユーザー名、パスワード）を入力し、「OK」をクリックします。入力されたデータを保存する前に、接続を検証するには、「**接続をテストする**」をクリックします。



7. 「OK」をクリックします。新規データソースがODBC データソースのリストに表示されます。

(C) 2015-2021 Altova GmbH



8. 「接続」をクリックします。

16.1.10.18 Sybase (JDBC) への接続

このサンプルでは、JDBC を使用して Sybase データベースサーバーに接続する手順について説明されています。

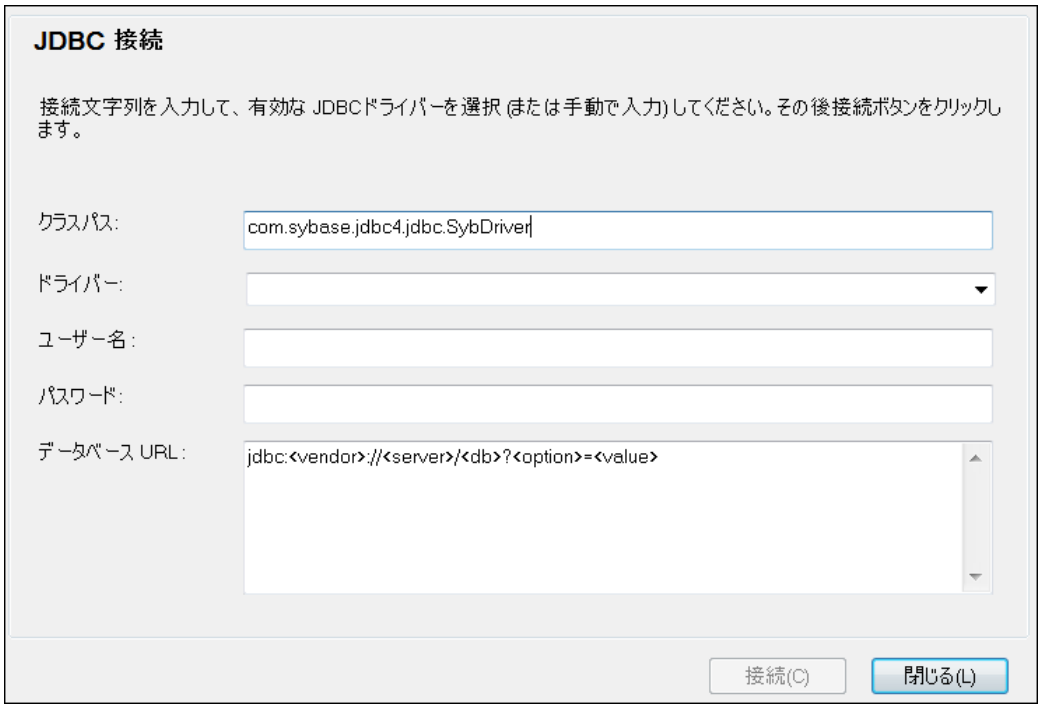
前提条件:

- JRE (Java Runtime Environment) または Java Development Kit (JDK) がインストールされている必要があります。これは Oracle JDK または Oracle OpenJDK などのオープンソースビルドであることができます。XMLSpy は次のロケーションから Java Virtual Machine (JVM) へのパスを次の順序で決定します: a) アプリケーション オプション内で設定したカスタム JVM パス b) Windows レジストリ内で見つかった JVM パス。c) `JAVA_HOME` 環境変数。
- XMLSpy (32-ビット、64-ビット) のプラットフォームが JRE/JDK のプラットフォームに一致することを確認してください。
- Sybase *jConnect* コンポーネントがオペレーティングシステムにインストールされている必要があります (このサンプルでは *jConnect 7.0* が使用されており、*Sybase Adaptive Server Enterprise PC Client* インストールの一部としてインストールされています)。データベースクライアントのインストール命令に関しては、Sybase ドキュメントを参照してください。
- 次のデータベース接続詳細が存在します: ホスト、データベース、ポート、ユーザー名、および、パスワード。

JDBC を使用して Sybase へ接続する:

1. データベース接続ウィザードを開始します。574
2. 「JDBC 接続」をクリックします。

3. 「クラスパス」の横にデータベースへの接続を与える .jar ファイルへのパスを入力します。必要であれば、セミコロンで区切られた .jar ファイルパスのリストを入力することもできます。このサンプルでは、必要とされる .jar ファイルは次のパスを使用して見つけることができます: **C:\sybase\jConnect-7_0\classes\jconn4.jar**。.jar ファイルパスをオペレーティングシステムの CLASSPATH 環境変数に追加した場合、「クラスパス」テキストボックスを空白のままにすることができます。(次も参照: CLASSPATH の構成 596)。
4. 「ドライバー」ボックスから、 を選択します。「ドライバー」ボックスから、**com.sybase.jdbc4.jdbc.SybDriver** を選択します。このエントリは、有効な .jar ファイルパスが、「クラスパス」テキストボックス内、または、オペレーティングシステムの CLASSPATH 環境変数内で検出された場合のみ使用することができます(前のステップを参照してください)。



5. データベースのユーザー名とパスワードを対応するテキストボックスに入力します。
6. ハイライトされた値をデータベースサーバーに適用することのできる値と置き換えて。データベース URL テキストボックス内のデータベースサーバーの接続文字列を入力します。

```
jdbc:sybase:Tds:hostName:port/databaseName
```

7. 「**接続**」をクリックします。

16.1.10.19 Teradata (JDBC) への接続

このサンプルでは、JDBC を使用した Teradata データベースへの接続方法について説明されています。

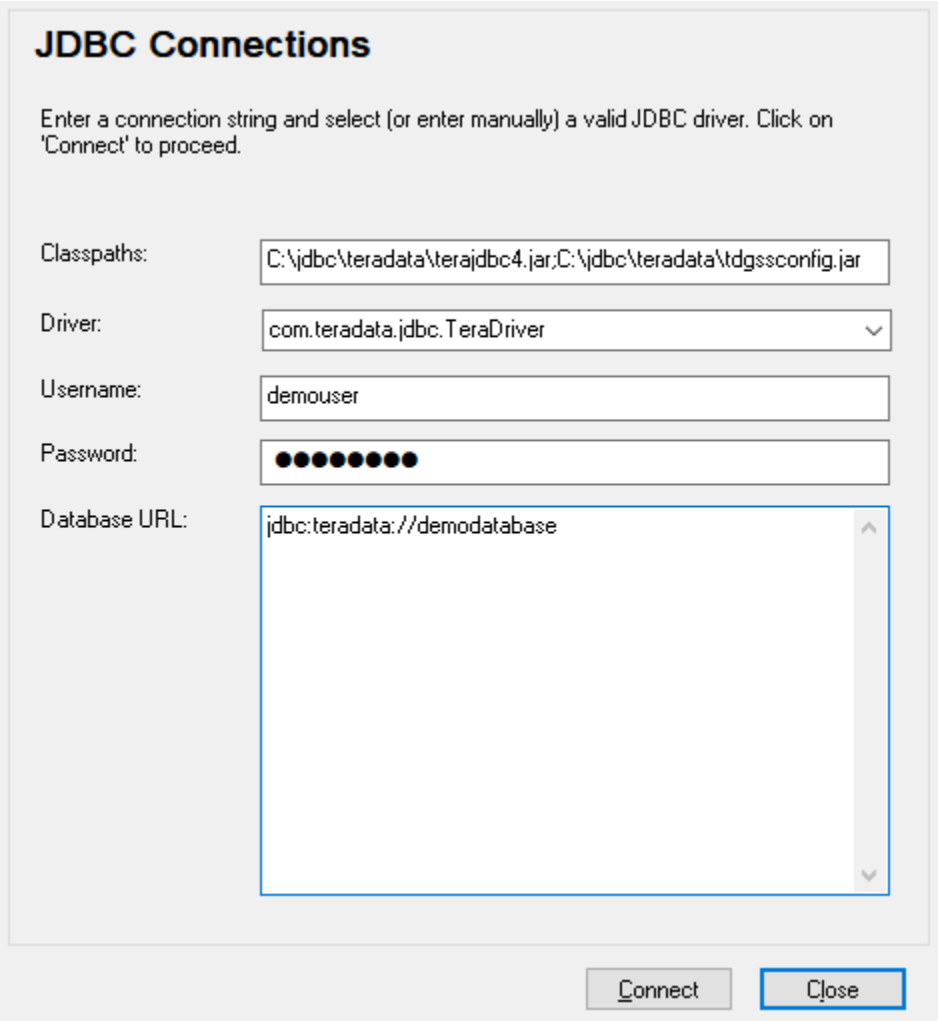
必要条件:

- JRE (Java Runtime Environment) または Java Development Kit (JDK) がインストールされている必要があります。これは Oracle JDK または Oracle OpenJDK などのオープンソースビルドであることができます。XMLSpy は次のロケーションから Java Virtual Machine (JVM) へのパスを次の順序で決定します: a) アプリケーションオプション内で設定したカスタム JVM パス b) Windows レジストリ内で見つかった JVM パス。c) `JAVA_HOME` 環境変数。
- XMLSpy (32-ビット、64-ビット) のプラットフォームが JRE/JDK のプラットフォームに一致することを確認してください。

- JDBC ドライバー（データベースへの接続を提供する1つまたは複数の.jar ファイル）がオペレーティングシステム上で使用できる必要があります。このサンプルでは、Teradata JDBC ドライバー16.20.00.02 が使用されます。詳細に関しては、次を参照してください：http://downloads.teradata.com/download/connectivity/jdbc-driver。
- 次のデータベース接続詳細が存在します：ホスト、データベース、ポート、ユーザー名、および、パスワード。

JDBC を使用した Teradata への接続の方法:

1. データベース接続ウィザードを開始します 574。
2. 「JDBC 接続」をクリックします。
3. 「クラスパス」の横に、データベースへの接続を与える.jar ファイルへのパスを入力します。必要な場合、セミコロンで区切られた.jar ファイルパスのリストを入力することができます。このサンプルでは、.jar ファイルは、次のパスの場所に存在します：**C:\jdbc\teradata**。オペレーティングシステムのCLASSPATH 環境変数に.jar ファイルパスが追加されている場合、「クラスパス」テキストボックス空のままにすることができます（次も参照してください：CLASSPATH の構成 596）.
4. 「ドライバー」ボックス内で、**com.teradata.jdbc.TeraDriver** を選択します。有効な.jar ファイルパスが「クラスパス」テキストボックスで、または、オペレーティングシステムのCLASSPATH 環境変数内に存在すると、使用することができます（前のステップを参照してください）。



5. ユーザー名とパスワードを対応するテキストボックスのデータベース内に入力します。
6. ハイライトされた値をデータベースサーバーと適用することのできる値を置換え、データベースURL テキスト内にデータベースサーバーに接続文字列を入力します。

```
jdbc:teradata://databaseServerName
```

7. 「接続」をクリックします。

16.1.10.20 Teradata (ODBC) への接続

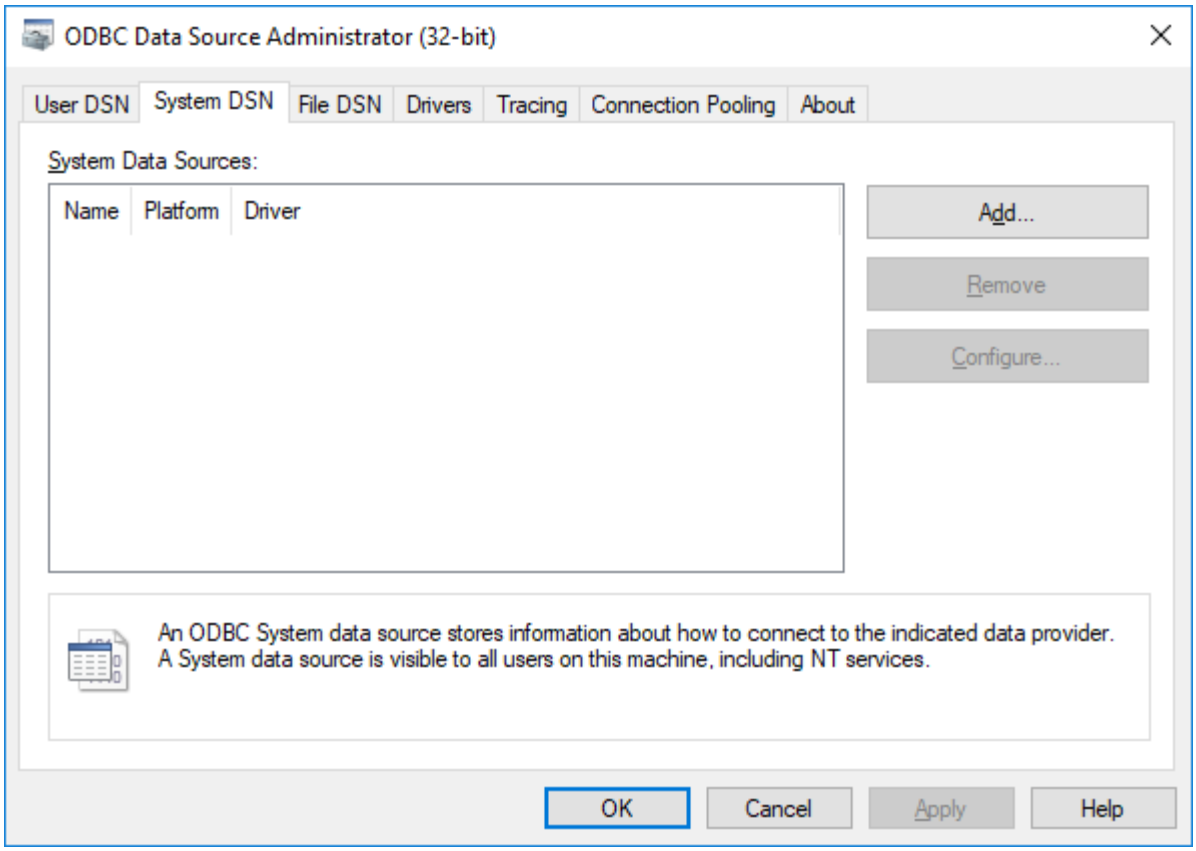
このサンプルでは、ODBC を使用した SQL Teradata データベースへの接続方法について説明されています。

前提条件:

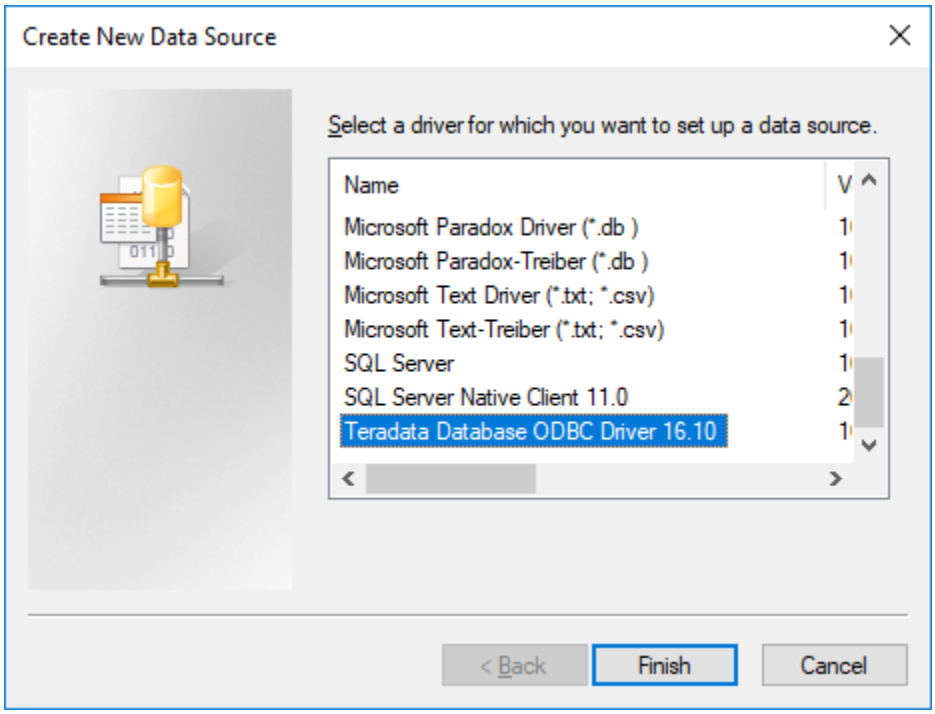
- Teradata ODBC ドライバーがインストールされる必要があります (参照: http://downloads.teradata.com/download/connectivity/odbc-driver/windows。このサンプルは、Windows バージョン 16.20.00 のために ODBC ドライバーを使用しています。
- 次のデータベース接続詳細が存在します: ホスト、ユーザー名、および、パスワード。

ODBC を使用した Teradata への接続の方法:

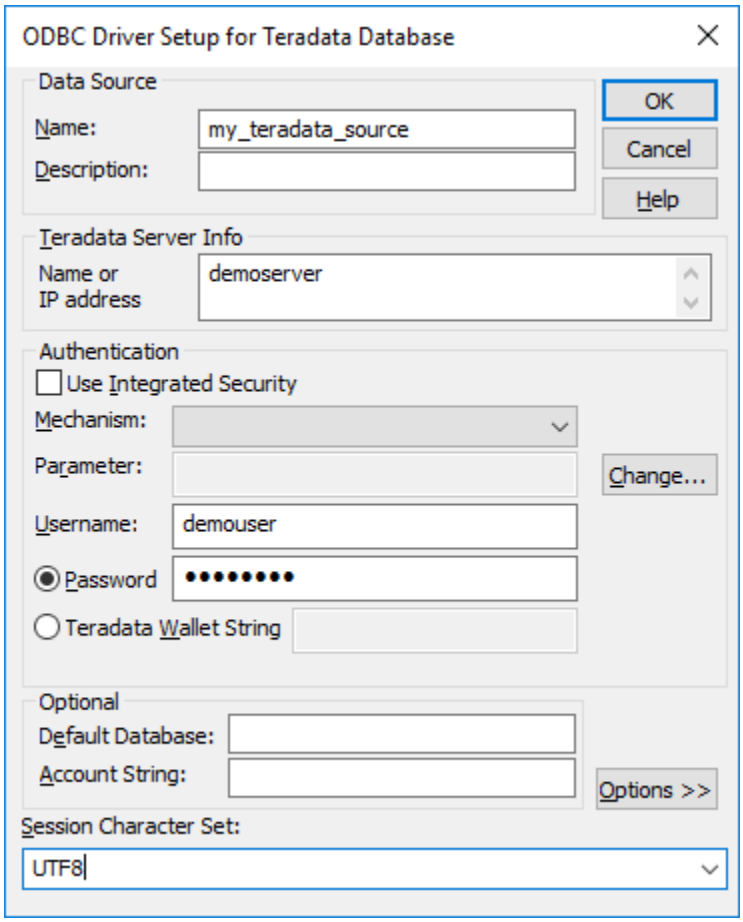
1. **Windows** キーを押して、「ODBC」を入力し、「ODBC データソースのセットアップ (32-ビット)」を提案されるリストから選択します。64-ビット ODBC ドライバーが存在する場合、「ODBC データソースのセットアップ (64-ビット)」を選択し、次のステップ内で 64-ビット XMLSpy を使用します。



2. 「System DSN」タブをクリックして、「追加」をクリックします。

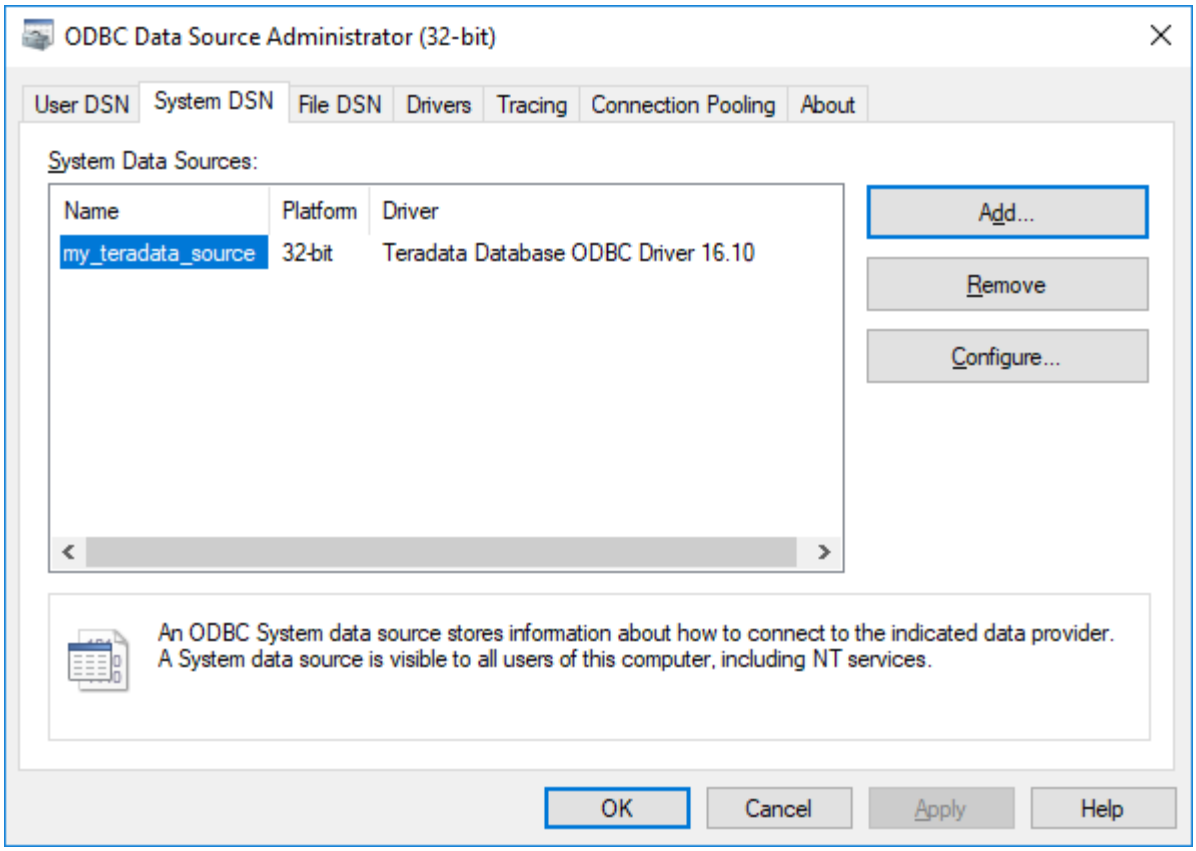


3. 「Teradata データベース ODBC ドライバー」を選択して、「終了」をクリックします。

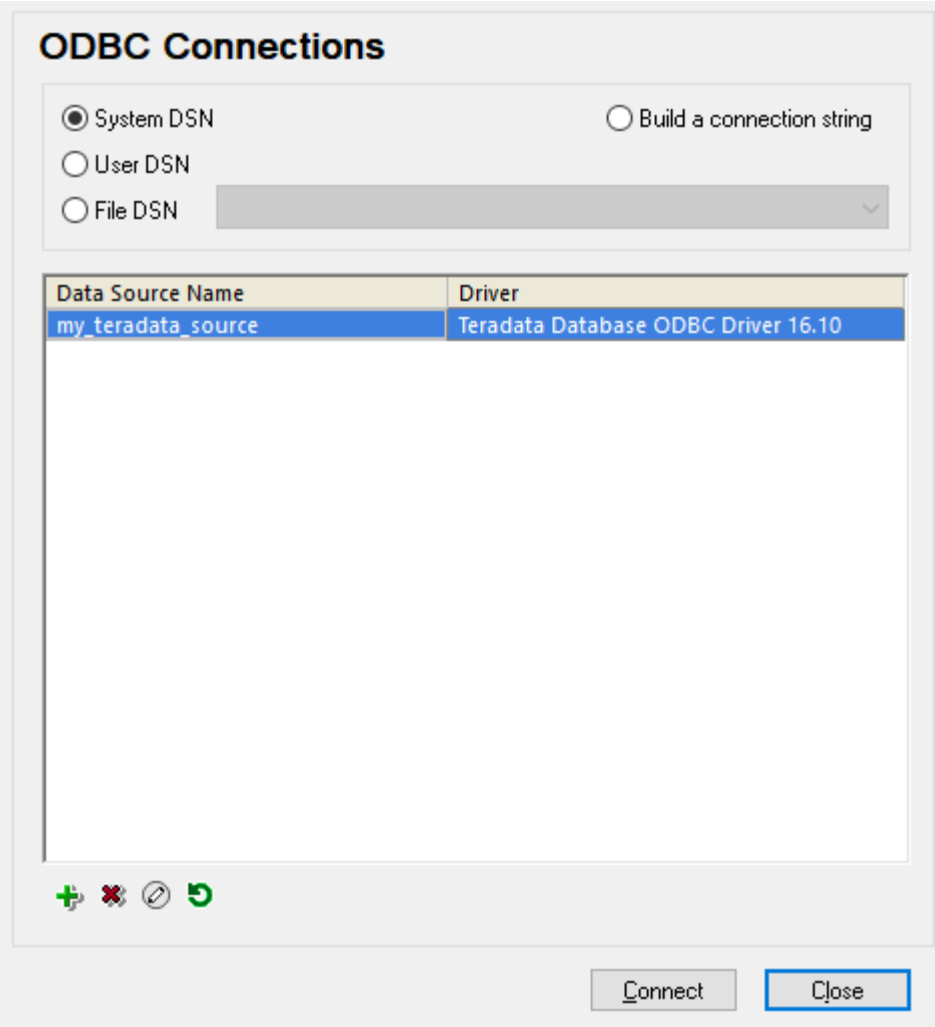


4. 名前を入力し、任意で、このODBC データソースを将来識別できる詳細を与えます。また、データベース接続資格情報（データベースサーバー、ユーザー、パスワード）を入力し、オプションでデータベースを選択します。
5. 「OK」をクリックします。データソースがリスト内に表示されます。

(C) 2015-2021 Altova GmbH



6. XMLSpy を作動し、データベース接続ウィザードを開始します[574]。
7. 「ODBC 接続」をクリックします。



8. 「**System DSN**」をクリックし、前に作成されているデータソースを選択し、「**接続**」をクリックします。

メモ　次のエラーが発生した場合: 「ドライバーが無効として返されました（または返しませんでした）SQL_DRIVER_ODBC_VER: 03.80」「ドライバーが無効と返されました」。ODBC クライアントへのパスが使用中のシステムのPATH 環境変数内に存在することを確認してください（例えば、このロケーションにインストールされている場合、**C:\Program Files\Teradata\Client\16.10\bin**）。このパスが不足している場合、手動で追加します。

(C) 2015-2021 Altova GmbH

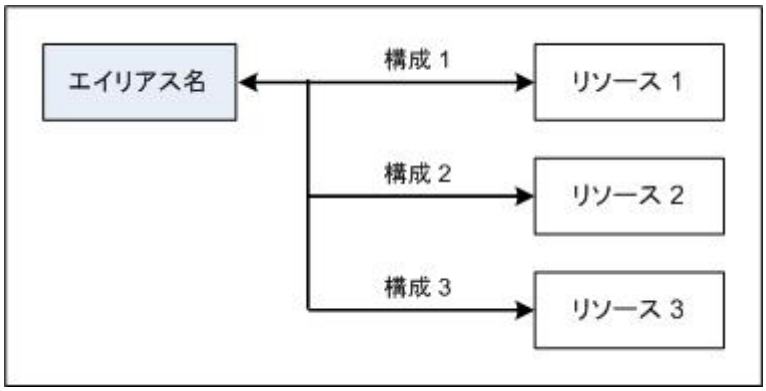
16.2 サポートされているデータベース

次のデータベースはサポートされています。各データベースの使用できるルート オブジェクトもリストされています。Altova は他の ODBC/ADO データベースへのサポートの向上のための努力をしていますが、以下にリストされたデータベースのみで接続とデータ処理のテストに成功しています。使用中のAltova アプリケーションが、64 ビット バージョンの場合、接続されているデータベース特定の64 ビット データベース ドライバーにアクセスしてください。Altova アプリケーションはIBM iSeries データベースのロジカル ファイルをサポートし、ロジカル ファイルを表示します。

データベース	メモ
Firebird 2.5.4, 3.0	
IBM DB2 8.x, 9.1, 9.5, 9.7, 10.1, 10.5	
IBM Db2 for i 6.1, 7.1, 7.2, 7.3, 7.4	ロジカルファイルはサポートされ、ビューとして表示されています。
IBM Informix 11.70, 12.10, 14.10	
MariaDB 10.2, 10.3, 10.4, 10.5	
Microsoft Access 2003, 2007, 2010, 2013, 2016, 2019	ドキュメントの作成時（2019 年9 月初旬）Access 2019 のために使用可能な Microsoft Access Runtime が存在しませんでした。Microsoft Access 2016 Runtime がインストールされており、データベースが「大量」のデータ型を使用しない場合のみ Altova 製品から Access 2019 にデータベース接続することができます。
Microsoft Azure SQL Database	SQL Server 2016 codebase
Microsoft SQL Server 2005, 2008, 2012, 2014, 2016, 2017, 2019 Microsoft SQL Server on Linux	
MySQL 5.0, 5.1, 5.5, 5.6, 5.7, 8.0	
Oracle 9i, 10g, 11g, 12c, 18, 19	
PostgreSQL 8.0, 8.1, 8.2, 8.3, 9.0.10, 9.1.6, 9.2.1, 9.4, 9.5, 9.6, 10, 11, 12.1	PostgreSQL 接続は、ODBC または JDBC などのインターフェイス（ドライバ）によりネイティブとドライバーベースの接続でサポートされています。ネイティブな接続はドライバを必要としません。
Progress OpenEdge 11.6	
SQLite 3.33.0	SQLite 接続は、ネイティブとしてサポートされます。SQLite データベースファイルへの直接の接続です。 別途のドライバーは必要ありません。Authentic ビューでは、SQLite からのデータを編集することはできません。Authentic ビューで、SQLite からのデータを保存する場合は、既知の制限についてのメッセージが通知されます。
Sybase ASE 15, 16	
Teradata 16	

17 Altova グローバルリソース

Altova グローバルリソースはファイル、フォルダー、そしてデータベースリソースのエイリアス集合のことです。各エイリアスは複数の構成を持ち、各構成を1つのリソースにマッピングすることができます（下のスクリーンショット参照）。そのため、グローバルリソースを入力として使用することで、グローバルリソースを構成とともに切り替えることができます。この操作は、アクティブな構成を選択することのできるGUI 上で簡単に行うことができます。例えば、XML ドキュメントを変換するXSLT スタイルシートがグローバルリソース（エイリアス）により割り当てられている場合、このグローバルリソースに対して複数の構成を作成し、各構成が異なるXSLT ファイルを指すようにすることができます。このようにグローバルリソースを設定すると、構成を切り替えることで変換に使用されるXSLT ファイルを変更することができます。



グローバルリソースはAltova 製品内部におけるリソースの切り替えだけでなく、他のAltova 製品から生成、使用されるリソースに対しても使用することができます。そのため、1つのAltova 製品から他のAltova 製品で使用されるファイルを一括して生成することができます。このような機能により、開発ならびにテストスピードが飛躍的に向上します。例えば、XMLSpy にあるXSLT スタイルシートを使って、Altova MapForce で生成されたファイルを変換することができます。

Altova グローバルリソースの使用には、2 つのステップがあります:

- グローバルリソースの定義 661: XML ファイルによりリソースが定義されます。これらのリソースは複数のAltova 製品間で共有することができます。
- グローバルリソースの使用 670: XMLSpy 内部ではファイルパスの代わりにグローバルリソースによりファイルを指定することができます。グローバルリソースを使用すると、XMLSpy にてアクティブな構成を切り替えることで、使用するリソースを簡単に変更することができるようになります。

その他のAltova 製品におけるグローバルリソース

現在のところ、以下のAltova 製品でグローバルリソースを使用することができます: XMLSpy、StyleVision、MapForce、Authentic Desktop、MobileTogether Designer およびDatabaseSpy。

(C) 2015-2021 Altova GmbH

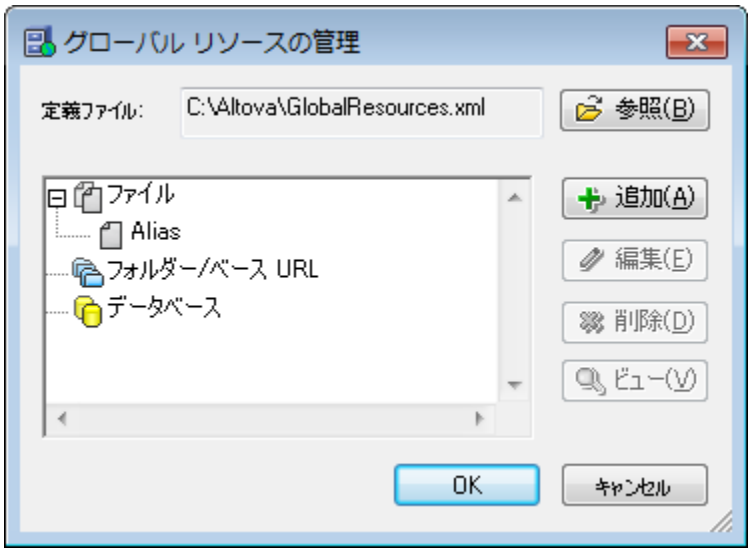
17.1 グローバル リソースの定義

Altova グローバルリソースは、以下の方法によりアクセスすることのできるグローバルリソースの管理ダイアログで定義されます:

- メニューコマンドの「**ツール | グローバルリソース**」を選択する。
- グローバルリソースツールバーにある「**グローバルリソースの管理**」アイコンをクリックする(下のスクリーンショット)。

グローバルリソース定義ファイル

グローバルリソースに関する情報はグローバルリソース定義ファイルと呼ばれるXML ファイルに収められています。このファイルは、グローバルリソースの管理ダイアログにて初めてグローバルリソースが定義ならびに保存された際に作成されます(下のスクリーンショットを参照)。



グローバルリソースの管理ダイアログを初めて開くと、グローバルリソース定義ファイルのデフォルトの場所と名前が、定義ファイルテキスト ボックスにて指定されます(上のスクリーンショットを参照):

```
C:\Users\<username>\My Documents\Altova\GlobalResources.xml
```

このファイルは、すべてのAltova アプリケーションに対するデフォルトのグローバルリソース定義ファイルとしてセットされます。従って、任意のAltova アプリケーションからこのファイルに対して保存されたグローバルリソースは、その他すべてのAltova アプリケーションにて直ちにグローバルリソースとして使用することができることになります。グローバルリソースを定義して、グローバルリソース定義ファイルに保存するには、グローバルリソースの管理ダイアログにてグローバルリソースを追加し、「**OK**」ボタンをクリックしてください。

既に存在するグローバルリソース定義ファイルを、実行中のAltova アプリケーションにおけるアクティブな定義ファイルとして選択するには、定義ファイルテキスト ボックスの隣にある「**参照**」ボタンから、そのファイルを選択してください。(上のスクリーンショットを参照)

メモ: グローバルリソース定義ファイルの名前は自由に変更することができ、Altova アプリケーションからアクセスできる任意の場所へ配置することができます。この場合、各アプリケーションにて(定義ファイルテキスト ボックスから) このファイルをアプリケーションのグローバルリソース定義ファイルとして指定する必要があります。複数のAltova 製品にて同一の定義ファイルを指定することで、これらのAltova 製品にてリソースが利用できるようになります。

メモ: 複数のグローバルリソース定義ファイルを作成することもできます。この場合、各 Altova アプリケーションにて同時に選択できるアクティブな定義ファイルは 1 つだけで、そのファイルの中に含まれている定義だけがアプリケーションでは利用できるということに注意してください。そのため、必要に応じて同じリソースを複数の定義ファイルで記述する必要が生じることもあります。

グローバルリソースの管理: 追加、編集、削除、保存

グローバルリソースの管理ダイアログでは、選択されたグローバルリソース定義ファイルへグローバルリソースを追加することができるほか、選択されたグローバルリソースを編集または削除することができます。ファイル、フォルダー、そしてデータベースのグループに追加されたグローバルリソースは、グローバルリソース定義ファイルにより管理されます(上のスクリーンショットを参照)。

グローバルリソースを追加するには、「追加」ボタンをクリックして、表示される「**グローバルリソース**」ダイアログにてグローバルリソースを定義します(このセクション以下のサブセクションにある「ファイル」663、「フォルダー」667、「データベース」661 の記述を参照ください)。グローバルリソースを定義して(グローバルリソースの管理ダイアログにある「OK」ボタンをクリックすることで)保存すると、選択されたグローバルリソース定義ファイルのグローバルリソース定義ライブラリに、そのグローバルリソースが追加されます。グローバルリソースはエイリアスにより識別されます。

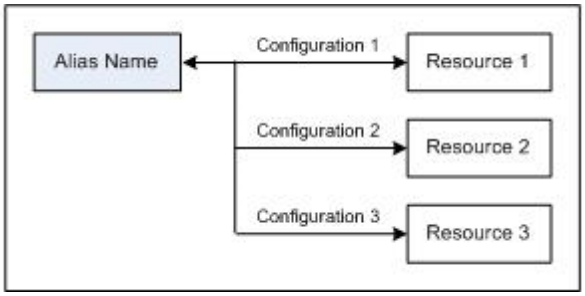
グローバルリソースを編集するには、「**編集**」ボタンをクリックします。関連するグローバルリソースダイアログが表示され、必要な変更を行うことができます(「ファイル」663、「フォルダー」667、「データベース」661 サブセクションの記述を参照ください)。

グローバルリソースを削除するには、「**削除**」ボタンをクリックします。

グローバルリソースの追加、編集、または削除を終えた後には、グローバルリソースの管理ダイアログにある「OK」ボタンをクリックして、グローバルリソース定義ファイルに**変更点を保存**するようにしてください。

構成によりグローバルリソースをエイリアス名へ関連付ける

グローバルリソースを定義すると、エイリアス名をリソース(ファイル、フォルダー、データベース)へマッピングすることになります。1 つのエイリアス名から複数のリソースへのマッピングを行うことができ、各マッピングは構成と呼ばれます。そのため、単一のエイリアス名から、複数の構成を介して複数のリソースへの関連付けを行うことができます(以下の図を参照)。



Altova アプリケーションではファイルの代わりにエイリアスを割り当てることができます。アプリケーションのアクティブなグローバルリソース構成を切り替えることで、各エイリアスに割り当てられたリソースを切り替えることができます。例えば Altova XMLSpy アプリケーションにて、MyXML.xml という名前の XML ドキュメントに対して XSLT 変換を行う場合、MyXSLT という名前のエイリアスを使用される XSLT としてグローバルリソースに割り当てることができます。XMLSpy では、アクティブな構成を変更することで、異なる XSLT ファイルを使用することができます。構成-1 にて First.xslt が MyXSLT へマッピングされており、構成-1 がアクティブな構成として選択されている場合、First.xslt が変換に使用されます。これにより、複数の構成を使用することで、単一のエイリアスから複数のリソースへアクセスすることができるようになります。このメカニズムはリソースのテストや比較を行う際に便利な機能となります。更に、グローバルリソースは複数の Altova 製品間で使用することができるため、複数の Altova 製品にてリソースのテストや比較を行うことができます。

 (C) 2015-2021 Altova GmbH

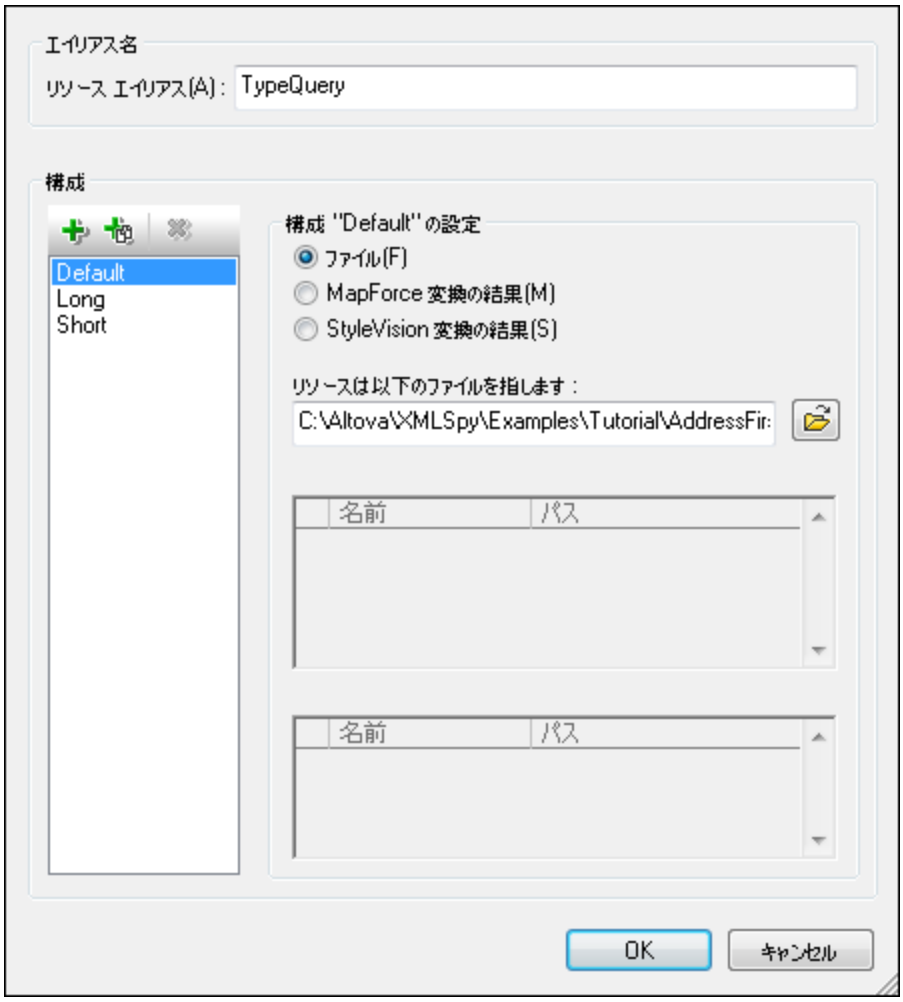
17.1.1 ファイル

「グローバルリソースの管理」661 ダイアログにある「追加 | ファイル」コマンドにより、ファイルのグローバルリソースダイアログへアクセスすることができます（以下のスクリーンショットを参照）。グローバルリソースダイアログでは、リソースエイリアステキスト ボックスにて入力されたエイリアスの構成を定義することができます。以下に記される方法により構成のプロパティを指定して、「OK」をクリックしてエイリアスの定義を保存してください。

エイリアスの定義を保存した後には上記の操作を（「グローバルリソースの管理」661 ダイアログにある「追加 | ファイル」コマンドを選択するところから）再度行うことで、別のエイリアスを追加することができます。

グローバルリソースダイアログ

エイリアスはグローバルリソースダイアログにて定義されます（以下のスクリーンショットを参照）。



グローバルリソースダイアログのアイコン

構成の追加: 追加する構成の名前を入力するための、構成の追加ダイアログが表示されます。

構成をコピーして追加: 選択された構成のコピーとして追加する構成の名前を入力するための、構成の追加ダイアログが表示されます。

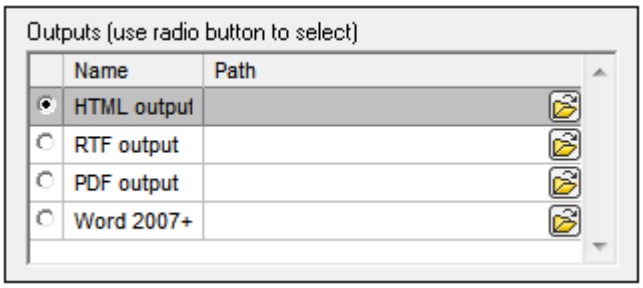
削除: 選択された構成を削除します。

開く: グローバルリソースとして作成されるファイルを参照します。

エイリアスの定義

以下の操作により、エイリアスの名前と構成を定義することができます:

1. *エイリアスに名前を付ける*: リソース エイリアス テキスト ボックスにエイリアスの名前を入力します。
2. *構成を追加する*: 構成ペインには、デフォルトで(名前変更や削除を行うことができない) Default という名前の構成が表示されます。(i) 構成の追加または構成のコピーを追加アイコンをクリックして、(ii) 表示されるダイアログにて**構成名を入力**することで、必要なだけの**構成を追加**することができます。追加された各構成は構成リストに表示されます。上のスクリーンショットでは、Long と Short という名前の構成が構成リストに追加されています。構成のコピーを追加コマンドにより、選択されている構成をコピーして、修正することもできます。
3. *各構成に対してリソースの種類を選択する*: 構成のリストから目的の構成を選択し、構成の設定ペインにて、(i) ファイル、(ii) MapForce 変換の結果、または(iii) StyleVision 変換の結果から、構成のリソースを指定します。対応するラジオ ボタンを選択してください。MapForce または StyleVision 変換の結果が選択された場合、.mfd または .sps ファイルと対応する入力ファイルを使って MapForce または StyleVision により変換が行われ、その変換結果がリソースとなります。
4. *リソースの種類に対するファイルを選択する*: リソースの種類にファイルが選択された場合、リソースのファイル テキスト ボックスにてファイルを選択してください。リソースの種類に変換の結果が選択された場合、ファイルの選択テキスト ボックスにて(MapForce 変換の場合は) .mfd ファイルまたは(StyleVision 変換の場合は) .sps ファイルを選択してください。変換に対して複数の入力または出力が可能な場合、選択するためのオプションが表示されます。例えば、StyleVision 変換の出力オプションが、インストールされている StyleVision のエディションに合わせて表示されます(以下のスクリーンショットでは、Enterprise Edition の出力が示されています)。



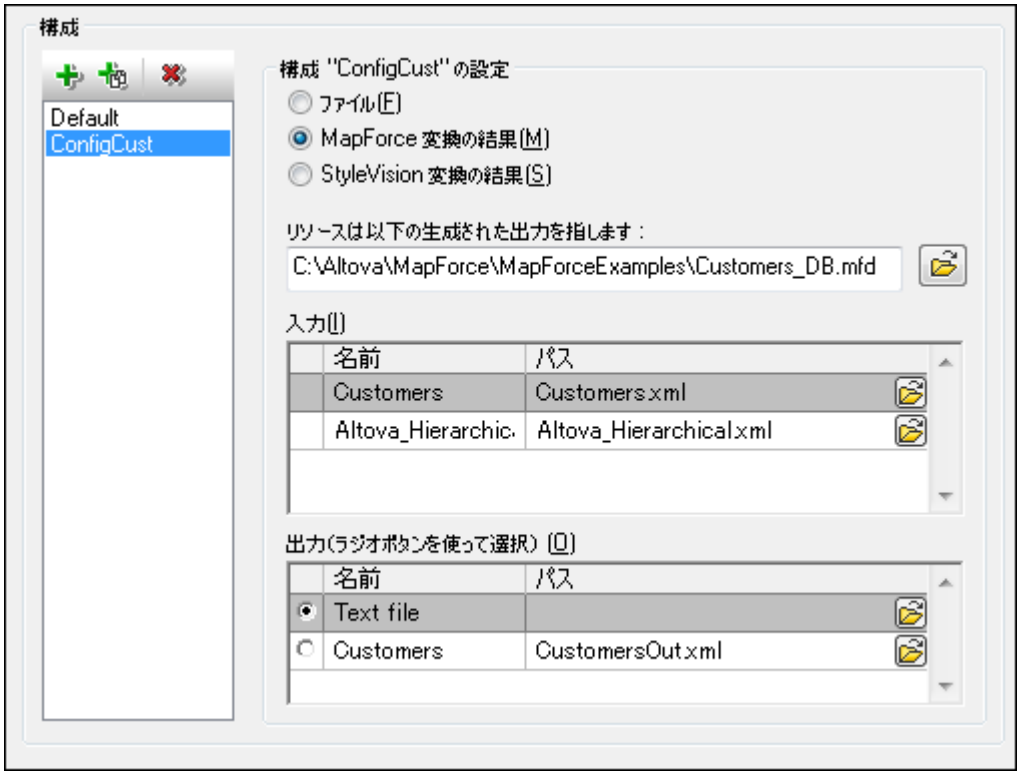
目的のオプションに対応するラジオ ボタンを選択します(上のスクリーンショットでは、「HTML 出力」が選択されています)。変換の結果をリソースとする場合、出力をファイルまたはグローバルリソースとして保存することができます。 アイコンをクリックして、「他のグローバルリソースを選択」または、「参照」を選択してください。どちらのオプションも選択されていない場合、グローバルリソースが使用された際に変換の結果が一時ファイルとしてロードされます。

5. *必要に応じて複数の構成を定義する*: その他の構成を必要な数だけ追加して、各構成に対してリソースを指定します。各構成に対して上記のステップ 3 と 4 を実行してください。エイリアスに対する新たな構成はいつでも追加することができます。
6. *エイリアスの定義を保存する*: 「OK」をクリックしてエイリアスと、すべての構成をグローバルリソースとして保存します。グローバルリソースが、「グローバルリソースの管理」ダイアログ 661 にあるファイル以下に表示されます。

MapForce 変換の結果

Altova MapForce では、(既存の) 入力ドキュメント スキーマから出力ドキュメント スキーマへのマッピングが行われます。MapForce ユーザーにより作成されたこのマッピングは、MapForce Design (MFD) と呼ばれます。入力スキーマに関連付けられた XML ファイル、テキスト ファイル、データベースは、データ ソースとして使用することができ、MapForce により出力ドキュメント スキーマに対応した出力データ ファイルが生成されます。「MapForce 変換の結果」が出力ドキュメントで、グローバルリソースになります。

MapForce により生成されたデータ ファイルをグローバルリソースとしてセットするには、グローバルリソース ダイアログにて、以下の設定を行う必要があります (以下のスクリーンショットを参照)。



- **.mfd (MapForce Design) ファイル**: 「リソースは以下の生成された出力を指します」テキスト ボックスにて、.mfd ファイルを指定する必要があります。
- **入力データ ファイル**: MFD ファイルが指定されると、そのファイルが解析され、中に含まれる入力スキーマ情報をベースにデフォルトのデータ ファイルが入力ペインに入力されます (上のスクリーンショットを参照)。他のファイルを指定することで、各入力スキーマに対するデフォルト ファイルの選択を変更することができます。
- **出力ファイル**: MFD ドキュメントに複数の出力スキーマが含まれる場合、出力ペインにそれらの出力すべてが表示されます (上のスクリーンショットを参照)。個々の出力スキーマに関連付けられた出力ファイルの場所が MFD ドキュメントにて指定されている場合、その出力スキーマに対するファイルの位置が出力ペインに表示されます。上のスクリーンショットからは、Customers 出力スキーマにデフォルトの XML データ ファイル (CustomersOut.xml) が与えられており、Text ファイル出力 スキーマにはそのような割り当てが行われていないことが理解できます。出力ペインに表示あるデフォルトのファイルの場所を使用することができるほか、自分で指定することができます。MapForce 変換の結果が、選択された出力スキーマがある場所へ保存され、そのファイルがグローバルリソースとして使用されます。

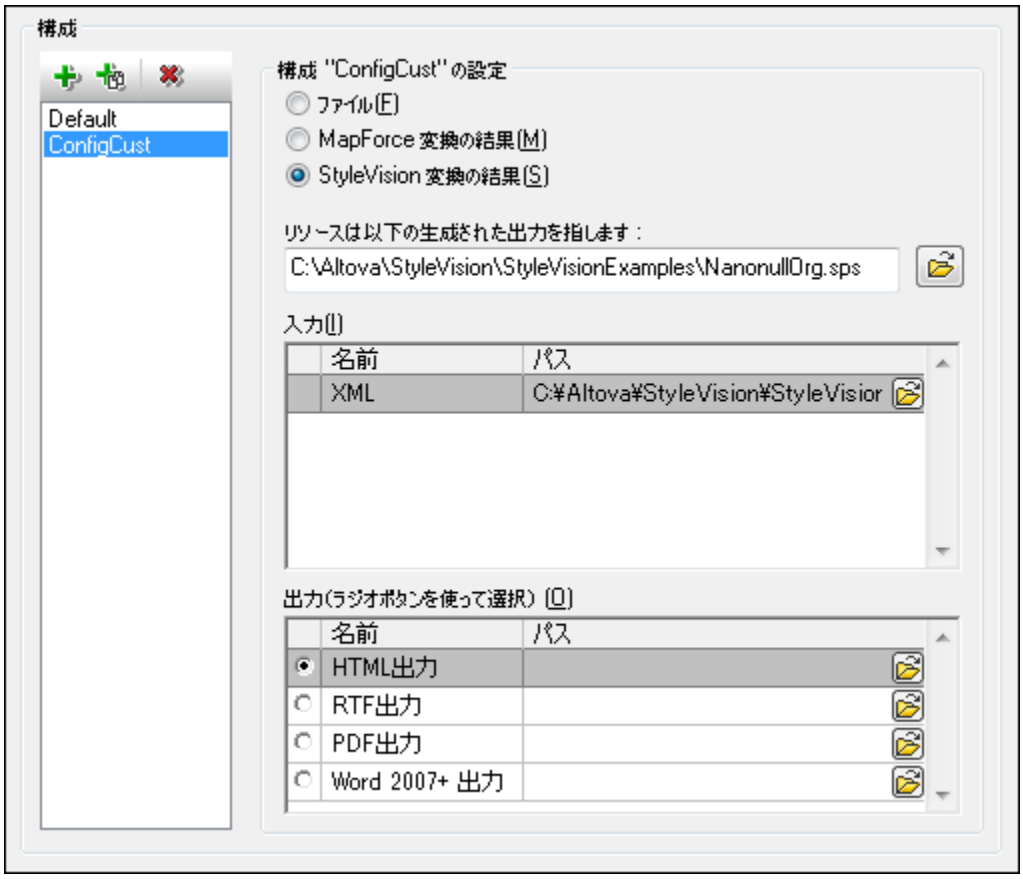
メモ　このオプション (MapForce の結果) により、グローバルリソースが呼び出された時に変換が行われるようになります。これで (入力ファイルから得られた) 最新のデータがグローバルリソースに含まれることが保証されます。

メモ 変換を行うためにはMapForce が使用されるため、この機能を使用するにはAltova MapForce をインストールする必要があります。

StyleVision 変換の結果

Altova StyleVision はStyleVision Power Stylesheet（SPS）ファイルを作成するために使用され、これらSPS ファイルにより、XML ドキュメントを様々なフォーマット（HTML、PDF、RTF、W ord 2007+ など）の出力ドキュメントへ変換するためのXSLT スタイルシートが生成されます。「StyleVision 変換の結果」オプションを選択すると、StyleVision により作成された出力ドキュメントが、選択された構成に関連付けられたグローバルリソースとなります。

グローバルリソースダイアログにて「StyleVision 変換の結果」オプションを選択すると、以下のファイルを指定する必要があります。



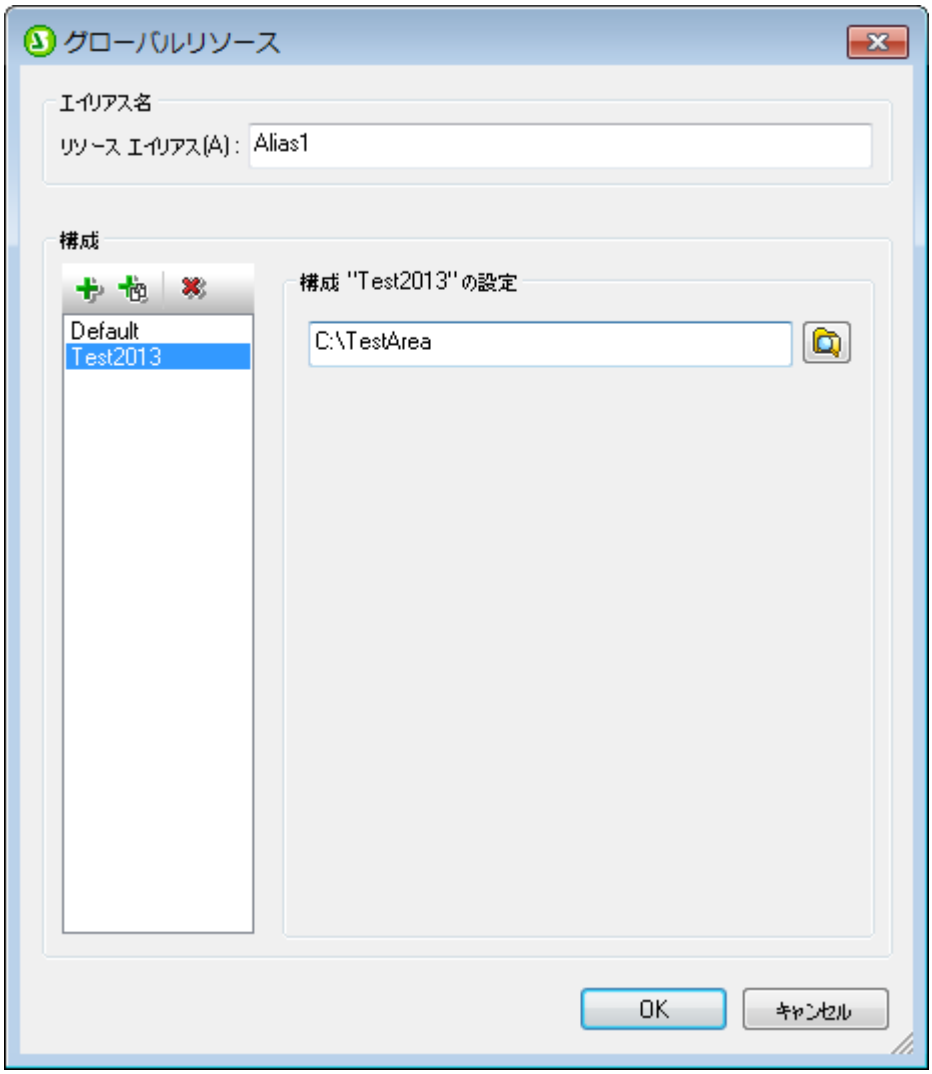
- ***.sps*** **（SPS）ファイル**: 「リソースは以下の生成された出力を指します」テキスト ボックスにて、ファイルを指定する必要があります（上のスクリーンショットを参照）。
- **入力ファイル**: 入力ファイルがSPS ファイル内部で既に指定されている場合、SPS ファイルが選択された段階で、そのファイルが入力ペインに表示されます。このエントリーは変更することができるほか、エントリーが指定されていない場合、追加する必要があります。
- **出力ファイル**: 出力ペインにて出力フォーマットを選択し、そのフォーマットに対する出力ファイルの場所を指定します。

メモ このオプション（StyleVision の結果）により、グローバルリソースが呼び出された時に変換が行われるようになります。これで、（入力ファイルから得られた）最新のデータがグローバルリソースに含まれることが保証されます。

メモ 変換を行うためにはStyleVision が使用されるため、この機能を使用するにはAltova StyleVision をインストールする必要があります。

17.1.2　フォルダー

フォルダーに対するグローバルリソースダイアログでは、以下の操作によりフォルダーリソースを追加することができます（以下のスクリーンショットを参照）。



グローバルリソースダイアログのアイコン

構成の追加: 追加する構成の名前を入力するための、構成の追加ダイアログが表示されます。

構成をコピーして追加: 選択された構成のコピーとして追加する構成の名前を入力するための、構成の追加ダイアログが表示されます。

削除: 選択された構成を削除します。

開く: グローバルリソースとして作成されるファイルを参照します。

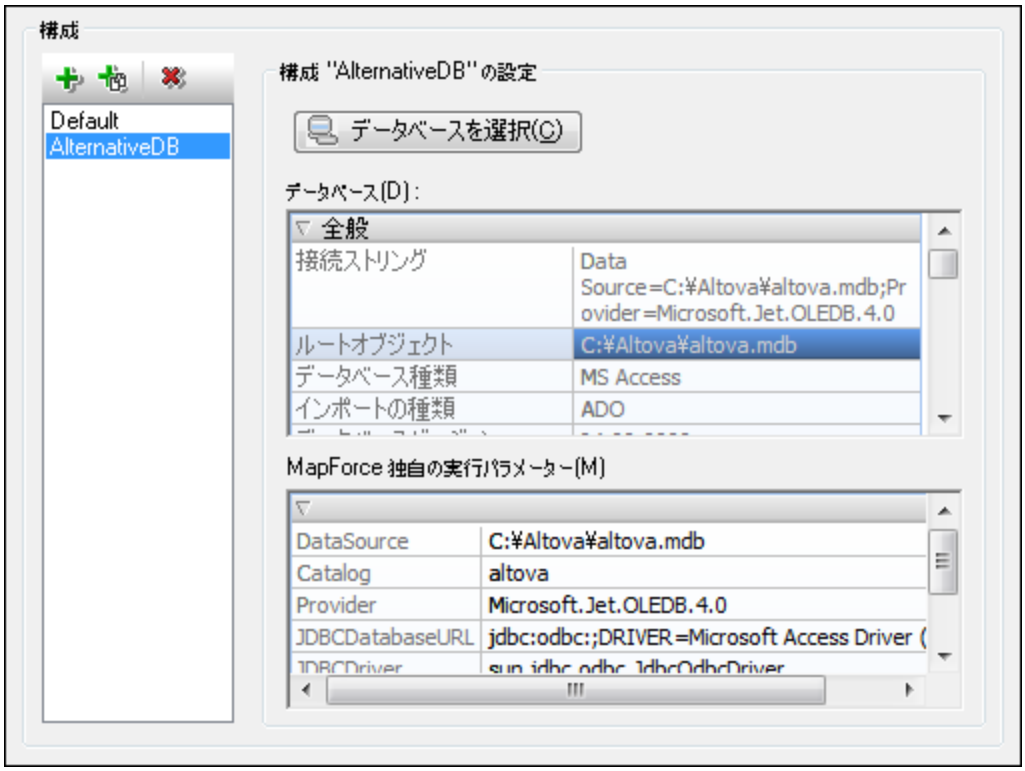
エイリアスの定義

以下の操作によりエイリアス(名前と構成) を定義することができます:

1. **エイリアスに名前を付ける:** リソース エイリアス テキスト ボックスにエイリアスの名前を入力します。
2. **構成を追加する:** 構成ペインには、デフォルトで(名前変更や削除を行うことができない) Default という名前の構成が表示されます。**構成の追加**または**構成のコピー**を追加アイコンをクリックして、表示されるダイアログにて構成名を入力することで、必要なだけの構成を追加することができます。構成の追加または構成のコピーを追加アイコンをクリックして、表示されるダイアログで構成の名前を入力することで、構成が追加されます。「OK」をクリックすることで新たな構成が構成ペインに表示されます。同じ作業を必要な数だけ繰り返してください。
3. **構成のリソースとなるフォルダーを選択する:** 構成ペインに表示された構成を選択し、グローバルリソースとして作成するフォルダーを参照します。セキュリティ資格情報がフォルダーにアクセスするために必要とされる場合、ユーザー名とパスワードフィールドでこれらを指定してください。フォルダーにアクセスするためにセキュリティの資格条件が必要な場合、ユーザー名とパスワードフィールドを指定してください。
4. **必要に応じて複数の構成を定義する:** その他の構成を必要な数だけ追加して、各構成に対してリソースを指定します。各構成に対して上記のステップ 2 と3 を実行してください。エイリアスに対する新たな構成はいつでも追加することができます。
5. **エイリアスの定義を保存する:** 「OK」をクリックしてエイリアスと、すべての構成をグローバルリソースとして保存します。グローバルリソースが、「グローバルリソースの管理」ダイアログ 661 にあるファイル以下に表示されます。

17.1.3 データベース

データベースに対するグローバルリソース ダイアログでは、以下の操作によりデータベースのリソースを追加することができます:



グローバル リソース ダイアログのアイコン

構成の追加: 追加する構成の名前を入力するための、構成の追加ダイアログが表示されます。

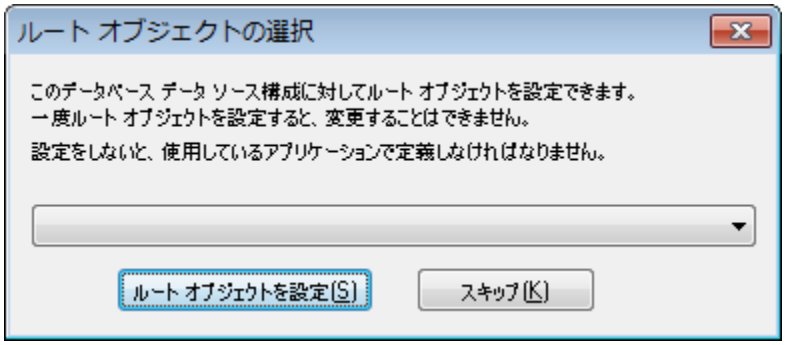
構成をコピーして追加: 選択された構成のコピーとして追加する構成の名前を入力するための、構成の追加ダイアログが表示されます。

削除: 選択された構成を削除します。

エイリアスの定義

以下の操作によりエイリアス(名前と構成) を定義することができます:

1. *エイリアスに名前を付ける*: リソース エイリアス テキスト ボックスにエイリアスの名前を入力します。
2. *構成を追加する*: 構成ペインには、デフォルトで(名前変更や削除を行うことができない) Default という名前の構成が表示されます。構成の追加または構成のコピーを追加アイコンをクリックして、表示されるダイアログにて構成名を入力することで、必要なだけの構成を追加することができます。構成の追加または構成のコピーを追加アイコンをクリックして、表示されるダイアログで構成の名前を入力することで、構成が追加されます。「OK」をクリックすることで新たな構成が構成ペインに表示されます。同じ作業を必要な数だけ繰り返してください。
3. *構成のリソースとなるデータベースを選択する*: 構成ペインに表示された構成を選択し、「**データベースを選択**」ボタンをクリックします。グローバルリソース接続を作成ダイアログが表示されます。フォルダーにアクセスするためにセキュリティの資格条件が必要な場合、ユーザー名とパスワードフィールドを指定してください。
4. *データベースへ接続する*: 接続ウィザード、既存の接続、 ADO 接続、ODBC 接続、または JDBC 接続を使用して、データベースへの接続を作成します。データベースへの接続[573] のセクションで説明されている接続メソッドの定義を完成します。接続ウィザード[574]、ADO 接続[578]、または、ODBC 接続[591] を使用することができます。 XMLSpy から既にデータベースへの接続が存在する場合は、既存の接続アイコンをクリックして、表示される接続リストからデータベースを選択します。
5. *ルート オブジェクトの選択*: ルート オブジェクトを選択することのできるデータベースへ接続した場合、サーバー上にあるルート オブジェクトを選択するためのダイアログが表示されます(以下のスクリーンショットを参照)。ルート オブジェクトを選択して、「**ルート オブジェクトを設定**」をクリックしてください。選択されたルート オブジェクトが、この構成が使用された時にロードされるルート オブジェクトとなります。



ルート オブジェクトを選択しなかった場合(「**スキップ**」ボタンをクリックした場合)、グローバルリソースがロードされた際にルート オブジェクトを選択することになります。

6. *複数の構成を必要に応じて定義する*: 作成されたその他の構成に対してデータベースリソースを指定します(作成された構成に対して上記のステップ 3 から 5 までを繰り返します)。新たな構成をエイリアス定義へ追加することもできます。
7. *エイリアスの定義の保存*: 「OK」をグローバルリソースダイアログ内でクリックし、エイリアスとその構成をグローバルリソースとして保存します。グローバルリソースは、グローバルリソースの管理ダイアログ内でデータベースの下にリストされています。

17.2 グローバルリソースを使用する

グローバルリソースには、ファイル型、フォルダー型、そしてデータベース型と、複数の種類が存在します。 XMLSpy で使用することのできる様々なシナリオについて以下で紹介されています: ファイルとフォルダー 670 および データベース 673 。

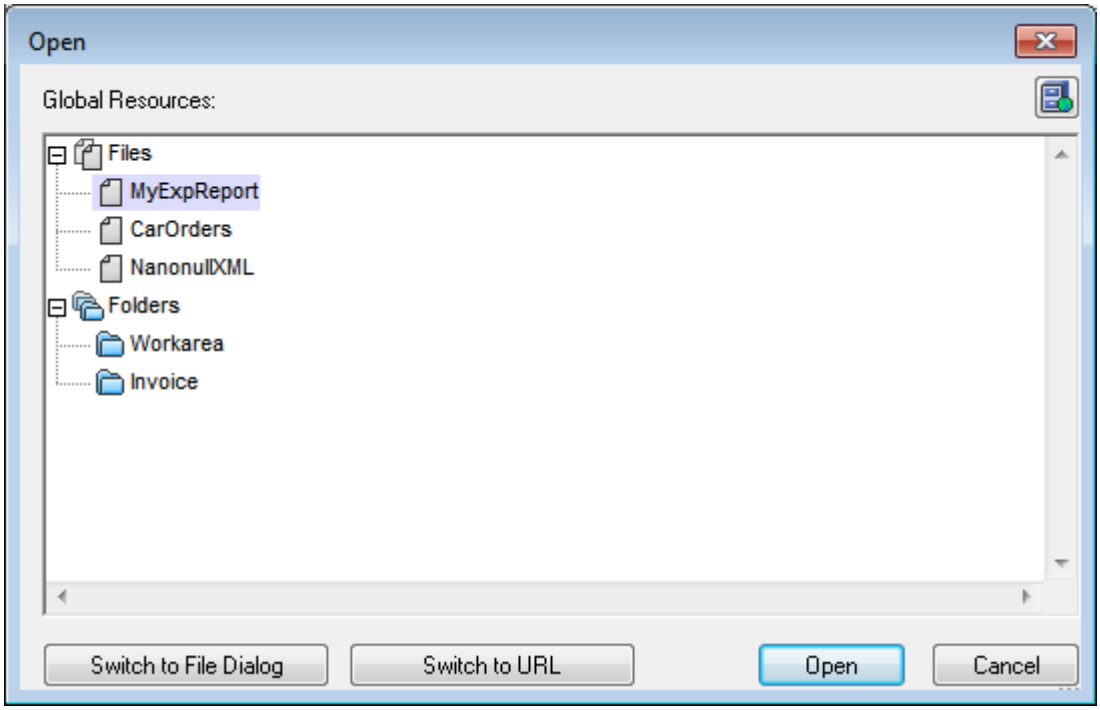
使用するリソースの選択

アプリケーション全体で使用することのできるグローバルリソースの選択、ならびに実際に使用されるグローバルリソースの選択を行うには方法が2つあります:

- 「グローバルリソース」ダイアログ 661 にてアクティブなグローバルリソース XML ファイルを選択します。 アクティブなグローバルリソース XML ファイル内に収められている利用可能なグローバルリソースの定義は、アプリケーション内の開かれているファイル全てで使用することができます。使用することができるのは、アクティブなグローバルリソース XML ファイルにある定義だけとなります。アクティブなグローバルリソース XML ファイルはいつでも変更することができ、新しいファイルに収められたグローバルリソース定義が、それまで使用されていたアクティブなファイルの内容に置き換えられます。従って、アクティブなグローバルリソース XML ファイルにより、(i) 割り当て可能なグローバルリソースと、(ii) 検索可能なグローバルリソースが決定されます(例えば、あるグローバルリソース XML ファイル内のグローバルリソースに割り当てが行われているが、現在アクティブなグローバルリソース XML ファイルにて、対応する名前のリソースがない場合、割り当てられたグローバルリソースの検索を行うことはできなくなります)。
- 「ツール | アクティブな構成」 994 またはグローバルリソース ツールバーからアクティブな構成 を選択します。メニューコマンド(またはツールバーのドロップダウンリスト)をクリックすることで、全てのエイリアスに対する構成のリストが表示されます。構成を選択することで、その構成がアプリケーションでアクティブな構成となります。この操作により、グローバルリソース(またはエイリアス)が使用される際には、各エイリアスに対して対応するアクティブな構成が適用されます。アクティブな構成は使用される全てのエイリアスに対して適用されます。アクティブな構成と同じ名前の構成をエイリアスが持たない場合、そのエイリアスに対してはデフォルトの構成が適用されます アクティブな構成はリソースの割り当てに影響を与えません。アクティブな構成は、実際にリソースが使用されるときに重要になります。

17.2.1 ファイルとフォルダーの割り当て

このセクションでは、ファイル型ならびにフォルダー型のグローバルリソース割り当て方法について記述します。ファイル型とフォルダー型のグローバルリソースは異なる方法で割り当てられます。以下にある使用シナリオでは、「**グローバルリソースに切り替え**」ボタンにてグローバルリソースを開くダイアログが表示されます(以下のスクリーンショットを参照)。



グローバルリソースの管理: 「グローバルリソースの管理」[661] ダイアログが表示されます。

ファイル型のグローバルリソースを選択することで、ファイルが割り当てられます。**フォルダー型のグローバルリソース**を選択することで、ファイルを開くダイアログが表示され、目的のファイルを選択することができます。選択されたファイルへのパスは、フォルダーリソースからの相対パスとなります。フォルダー型のグローバルリソースに2つの構成が含まれ、各構成が異なるフォルダーを指す場合（そして、それらのフォルダー内にあるファイルの名前が同一の場合）、構成切り替えることで2つの異なるフォルダーをターゲットにすることができます。これは試験などを行う際に便利な機能となります。

ダイアログ下部にあるボタンをクリックすることで、ファイルダイアログまたはURL ダイアログに切り替えることもできます。「**グローバルリソースの管理**」アイコンをクリックすることで、「グローバルリソースの管理」[661] ダイアログが表示されます。

使用できるシナリオ

ファイル型やフォルダー型のグローバルリソースに対して以下のような操作を行うことができます:

- グローバルリソースを開く[671]
- グローバルリソースとして保存[672]
- XSLT 変換にファイルを割り当てる[672]
- XSLT 変換[673]
- XQuery の実行[673]
- SPS の割り当て[673]

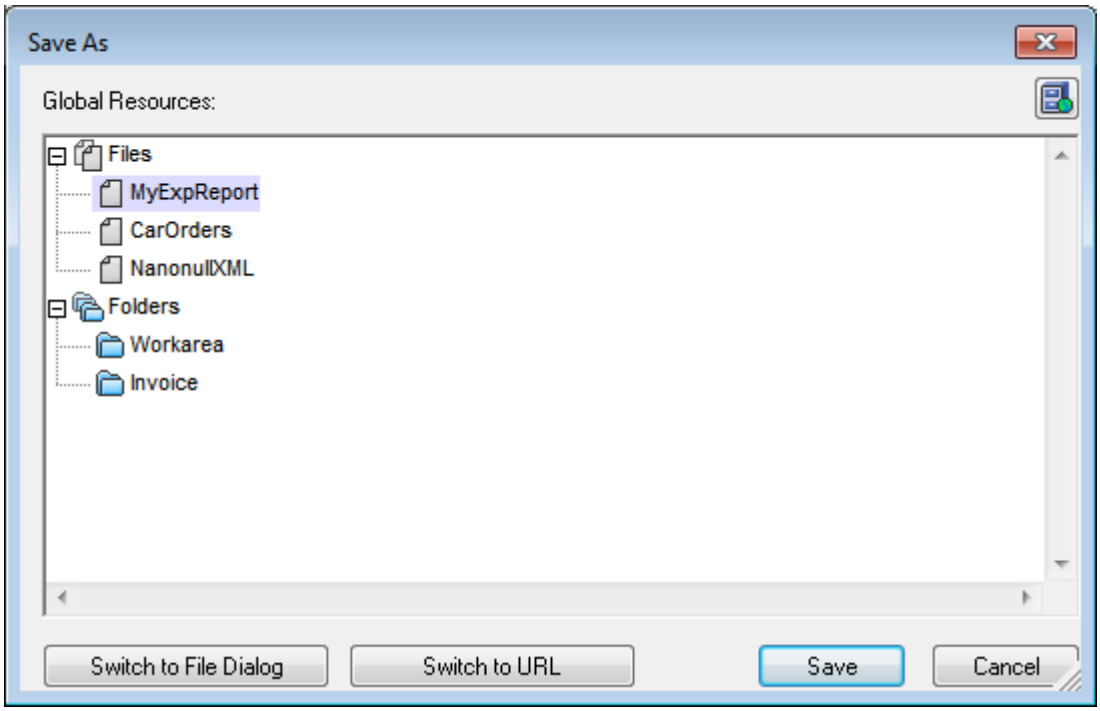
グローバルリソースを開く

グローバルリソースはXMLSpy の「ファイル | 開く（グローバルリソースに切り替え）|」[746] コマンドから開き、編集することができます。ファイル型グローバルリソースの場合、ファイルが直接開かれます。フォルダー型グローバルリソースの場合、割り当てられたフォルダーが選択された状態でファイルを開くダイアログが表示され、フォルダー以下にある目的のファイルを選択することになります。グローバルリソースを通してファイ

ルを選択し、編集する利点は、構成を切り替えるだけで、同一のグローバルリソース内にある(別構成の)ファイルへアクセスできることです。構成を変更する前に、編集がなされたファイルを保存を行う必要があります。

グローバルリソースとして保存

新たに作成されたファイルもグローバルリソースとして保存することができます。同様に、既存のファイルを開いて、グローバルリソースとして保存することもできます。「**ファイル | 保存**」または「**ファイル | 名前を付けて保存**」コマンドをクリックすると、保存ダイアログが表示されます。「**グローバルリソースに切り替え**」ボタンをクリックして、現在使用されているグローバルリソース XML ファイルにて定義されたエイリアスであるグローバルリソースへアクセスします(以下のスクリーンショットを参照)。



エイリアスを選択し、「**保存**」をクリックします。エイリアスがファイルエイリアス[663]の場合、そのファイルに直接保存されます。エイリアスがフォルダーエイリアス[667]の場合、ダイアログが表示され、保存するファイルの名前を入力するよう促されます。どちらの場合でも、ファイルは現在アクティブな構成[674]により定義された位置に保存されます。

メモ 各構成ではファイルの位置が構成の定義により指定されます。グローバルリソースとして保存しようとするファイルが、構成で定義された位置にあるファイルと同一の拡張子を持たない場合、このグローバルリソースが開かれたときに、XMLSpy にて編集エラーや検証エラーが発生する可能性があります。これは構成の定義により指定されたファイルの種類を想定して XMLSpy によりファイルが開かれるためです。

XSLT 変換にファイルを割り当てる

グローバルリソースを介して XSLT ファイルを XML ドキュメントに、XML ファイルを XSLT ドキュメントに割り当てることができます。XSLT ファイルを割り当てるコマンド「**XSL/XQuery | XSL の割り当て**」[894]ならびに「**XSL/XQuery | XSL:FO の割り当て**」[894]や XML ファイルを割り当てるコマンド「**XSL/XQuery | サンプル XML の割り当て**」[894]がクリックされると、割り当てダイアログが表示されます。「**参照**」ボタンをクリックすることで、ファイルを開くダイアログが表示され、グローバルリソースを開くダイアログに切り替え、目的のグローバルリソースを選択することができます。グローバルリソースを使用することで、XSLT 変換を行うために指定した XSLT(または XML)ファイルを、XMLSpy の構成を切り替えるだけで変更することができます。異なるファイルに対して変換が必要になった時も、ファイルの割り当て変更を行う必要はありません。XSLT 変換が開始されると、アクティブな構成で指定されるファイルが使用されます。

 (C) 2015-2021 Altova GmbH

XSLT 変換 ならびにXQuery の実行

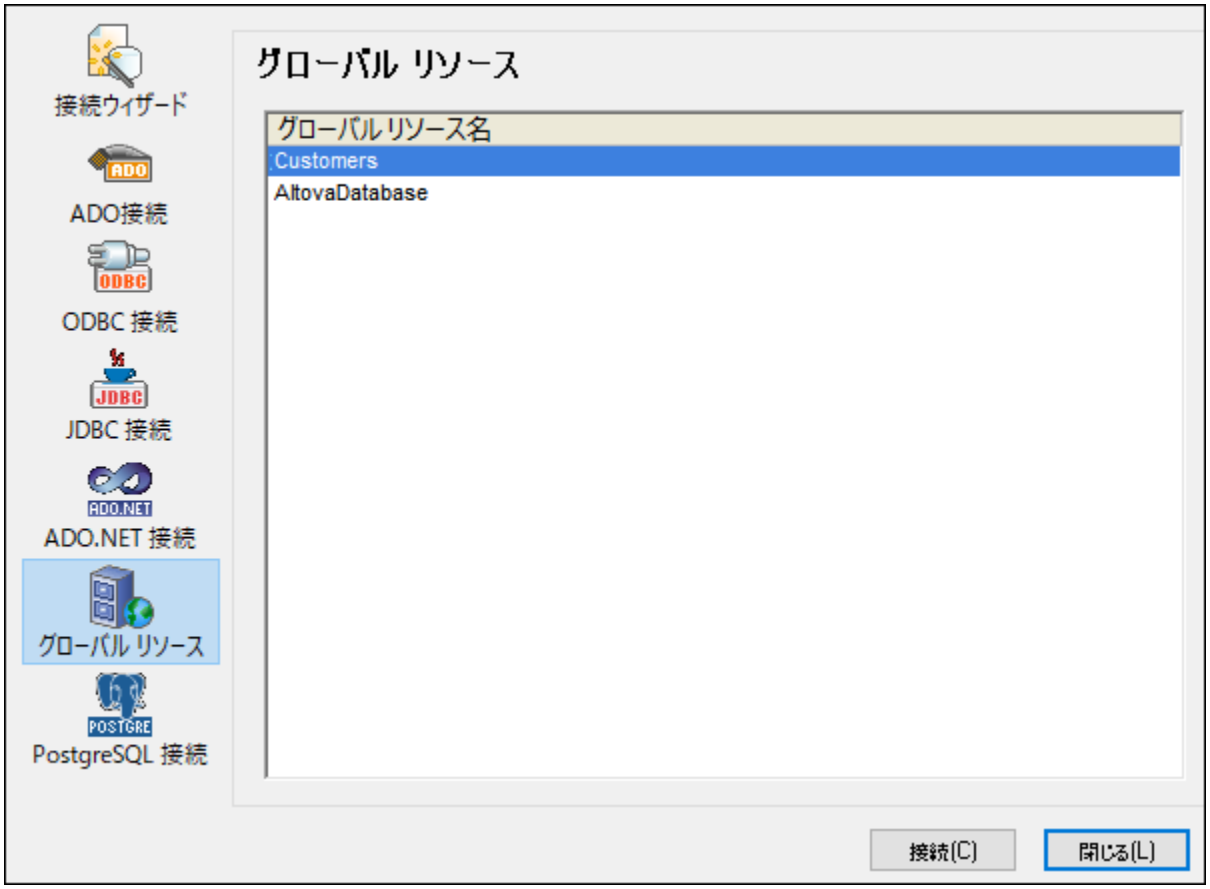
コマンド「XSL/XQuery | XSL 変換」[888] または「XSL/XQuery | XSL:FO 変換」[888] または「XSL/XQuery | XQuery 実行」[893] をクリックすると、XQuery または XML ファイルを指定するためのダイアログが表示されるので、、XSLT、XQuery、または XML ファイルを選択します。「参照」ボタンをクリックして、「グローバルリソースに切り替え」ボタンをクリックすることで、グローバルリソースを開くダイアログが表示されます(セクション最上部のスクリーンショットを参照[670])。選択されたグローバルリソースの現在アクティブな構成に関連付けられているファイルが変換に使用されます。

SPS の割り当て

StyleVision スタイルシートを XML ファイルに割り当てる際「Authentic | SPS ファイルの割り当て」、スタイルシートを指定するためにグローバルリソースを使用することができます。「参照」ボタンをクリックして、グローバルリソースリソースに切り替えボタンをクリックすることで、グローバルリソース開くダイアログが表示されます(セクション最上部のスクリーンショットを参照[670])。グローバルリソースを割当先として選択することで、XMLSpy のアクティブな構成を切り替えるだけで、XML ドキュメントの Authentic View を変更することができます。

17.2.2　データベースの割り当て

DB から XMLSpy へ、データや(XML スキーマとして)データ構造をインポートするコマンドを実行すると(例えば、「変換 | データベースのデータをインポート」コマンドを使用)、オプションとしてグローバルリソースを使用することができます(以下のスクリーンショットを参照)。メニューにあるデータベースに関連するコマンドでも、データベース型のグローバルリソースを使用することができます。メニューにあるデータベースに関連するコマンドでも、データベース型のグローバルリソースを使用することができます。



データベースを開くダイアログにてグローバルリソースアイコンをクリックすると、現在アクティブなグローバルリソース XML ファイル[661]に定義されている全てのデータベース型グローバルリソースが表示されます。目的のグローバルリソースを選択して、「**接続**」をクリックします。選択されたグローバルリソースに2つ以上の構成が割り当てられている場合、現在アクティブな構成に対応したグローバルリソースが使用され(「**ツール | アクティブな構成**」またはグローバルリソースツールバーを確認してください)、接続が確立されます。DB から XML スキーマを作成[946]や DB データのインポート[941]に記述されているように、使用するデータ構造やデータを選択する必要があります。

17.2.3 構成の切り替え

同時にアクティブにすることのできる構成は1つだけで、その設定はアプリケーション全体に対して適用されます。これは、アクティブな構成が、現在開いている全てのファイルやデータソース接続のエイリアスに対して有効であることを意味します。エイリアスにアクティブな構成と同じ名前の構成が割り当てられていない場合、デフォルトの構成がそのエイリアス対して使用されます。構成を変更する例として、XSLT ファイルが複数の構成を持つグローバルリソースを介して XML ドキュメントに割り当てられたと仮定しましょう。各構成は異なるファイルにマップされます。ですから、ファイルの選択はアプリケーションのアクティブな構成としてどの構成が選択されているかにより決まります。

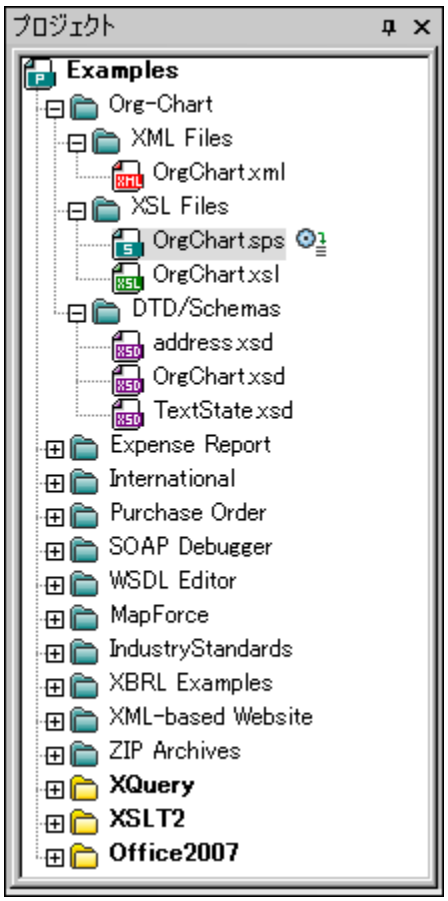
構成の切り替えるには2つの方法があります:

- メニューコマンド「**ツール | アクティブな構成**」を選択すると、グローバルリソース XML ファイル内にある全ての構成がサブメニューに表示されます。目的の構成を選択します。
- グローバルリソースコンボボックスのツールバーから、目的の構成を選択します(以下のスクリーンショットを参照)。

アクティブな構成を切り替えるという方法により、グローバルリソースを介して割り当てられたソースファイルを変更することができます。

18 プロジェクト

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



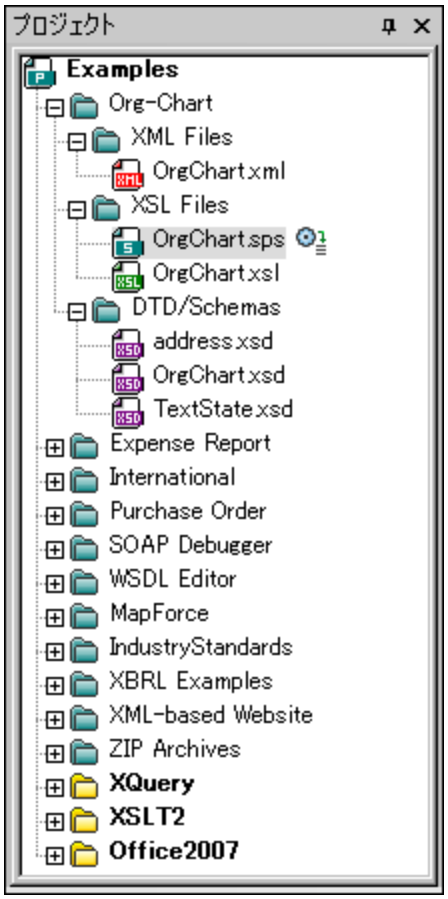
プロジェクトを使用することで、一緒に使用されるファイルをまとめて、ファイルへのアクセスを素早く行うことができます。更に、各フォルダーに対してスキーマやXSLT ファイルを定義することもでき、フォルダーに対するバッチ処理を行うこともできます。

このセクションではプロジェクトの作成と編集方法 677 ならびにプロジェクトの使用方法 681 を説明します。

(C) 2015-2021 Altova GmbH

18.1　プロジェクトの作成と編集

プロジェクトは(以下のスクリーンショットにある)プロジェクトウィンドウ[105]とプロジェクトメニュー[782]により管理されます。プロジェクトは一度に1つしか開くことができません。開かれたプロジェクトはプロジェクトウィンドウ[105]に表示されます。



プロジェクトの新規作成と既存プロジェクトを開く

メニューコマンド「**プロジェクト | 新規プロジェクトの作成**」により新たなプロジェクトが作成されます。既存のプロジェクトは「**プロジェクト | プロジェクトを開く**」メニューコマンドにより開くことができます。新たに開かれたプロジェクト(新規プロジェクトか既存のプロジェクトかに関わらず)は、それまでプロジェクトウィンドウで開かれていたプロジェクトを置き換えます。それまで開かれていたプロジェクトに(下のスクリーンショットにあるように、フォルダーの隣のアスタリスクとして示される)保存されていない変更が含まれている場合、変更を保存するかどうかを決定することになります。

プロジェクトの名前付けと保存

新規プロジェクトの名前は保存時に決定されます。プロジェクトの保存には、「**プロジェクト | プロジェクトの保存**」コマンドを使用し、プロジェクトは .spp というファイル拡張子で保存されます。プロジェクトが変更されると、その変更を反映するためにプロジェクトを保存しなければなりません。(プロジェクトウィンドウのトップレベルにある)プロジェクトの名前は、Windows のファイルシステムを使ってしか変更することができず、XMLSpy のGUI を使っての名前変更はできないことに注意してください。サブフォルダーの名前はGUI により変更することができます。

プロジェクトの構成

プロジェクトはフォルダーとファイルから成るツリー構造により管理されます。フォルダーとファイルはどの階層にも作成することができ、その深さにも制限はありません。サブフォルダーやファイル、またはリソースを作成するには、プロジェクトウィンドウ内のフォルダーを選択し、「**プロジェクト**」メニューにあるコマンド、またはコンテキストメニューを使用します。プロジェクトに追加されたファイル、フォルダー、そしてリソースを削除し、ドラッグによりプロジェクトツリー内の他の場所に移動することができます。

新規プロジェクトが作成されると、ファイルをその種類(XML やXSL など)により管理するデフォルトの構造が表示されます(以下のスクリーンショットを参照)。

フォルダーに定義されたプロパティにより、各フォルダーにはファイル拡張子が割り当てられています。ファイルがフォルダーに追加されると、そのファイル拡張子に従って、ファイルは自動的に対応するフォルダーに追加されます。各フォルダーに対して、対応するファイル拡張子を割り当てることができます。

プロジェクトに追加することのできるもの

フォルダー、ファイル、そしてその他のリソースは、トップレベルのプロジェクトフォルダーまたはプロジェクト内にある任意の階層のフォルダーに追加することができます。フォルダーには、(i) プロジェクトフォルダー、(ii) 外部フォルダー、(iii) 外部Web フォルダー、の3種類あります。

オブジェクトを追加するには、関連するフォルダーを選択し、必要なコマンドをそのフォルダーのコンテキストメニュー、または「**プロジェクト**」メニューから選択します。以下のオブジェクトをプロジェクトフォルダーに追加することができます:

- *プロジェクトフォルダー(緑色)* はプロジェクトのコンテンツを構造化するためにプロジェクトに追加されるフォルダーです。プロジェクトフォルダーには(そのフォルダーのプロパティから)対応するファイルの拡張子を割り当てることができます。ファイルがフォルダーに追加されると、そのファイルの拡張子にマッチした最初のフォルダー内に配置されます。従って、複数のファイルがフォルダーへ追加されると、それぞれのファイルの拡張子に従って、対応するフォルダーに配置されることになります。
- *外部フォルダー(黄色)* はファイルシステムのフォルダーです。外部フォルダーがプロジェクトに追加されると、その外部フォルダー内にある全てのファイル、サブフォルダー、そしてサブフォルダーのファイルがプロジェクトに追加されます。外部フォルダーに対してファイル拡張子を定義すると、そのフォルダー内のファイルをフィルタリングする効果を持ちます。
- *外部Web フォルダー*は外部フォルダーに似ていますが、Web サーバー上にあるフォルダーを指定し、アクセスのためにユーザー認証を行う必要があります。外部のWeb フォルダーに対してファイル拡張子を定義すると、そのフォルダー内のファイルをフィルタリングする効果を持ちます。
- *ファイル*はフォルダーの選択を行ない、以下の3つのファイル追加コマンドのどれかを選択することでフォルダーに追加することができます: (i)「**プロジェクトにファイルを追加**」コマンドにより開かれるダイアログにて追加するファイルを選択するか、(ii)「**プロジェクトにアクティブなファイルを追加**」コマンドによりメインウィンドウでアクティブなファイルをプロジェクトに追加する、または(iii) プロジェクトに「**アクティブならびに関連するファイルを追加**」コマンドによりアクティブなXML ファイルに関連するファイル(例えばXML スキーマやDTD)も一緒にプロジェクトにファイルを追加します。(例えばXSLT ファイルのような)処理命令により割り当てられているファイルは関連するファイルとは見なされないことに注意してください。
- *グローバルリソース*はファイル、フォルダー、そしてデータベースリソースへのエイリアスです。その定義方法と使用方法については、グローバルリソース 660 のセクションにて記述されます。*URL* を使えばURL によりオブジェクトを指定することができます。

- *Altova スクリプトプロジェクト* は.asprj 拡張子を持つファイルで、XMLSpy プロジェクトに関連付けることができます。これによりマクロやその他のスクリプトがプロジェクトで利用できるようになります。スクリプトプロジェクトの作成方法とXMLSpy プロジェクトへの割り当て方法はスクリプティング[1056]のセクションを参照ください。

Project and folder properties

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

フォルダーのプロパティはフォルダーを選択してメニューコマンド [**プロジェクト | プロパティ**] (またはフォルダーのコンテキストメニューコマンドプロパティ)を使用してアクセスするそのフォルダーのプロパティダイアログに保管されています。 フォルダーの以下のプロパティはプロパティダイアログ内で定義し編集可能です:

- *フォルダー名* :トップレベルのプロジェクトフォルダーに対しては編集できません(フォルダー名の代わりにファイルパスが表示されます)。
- *ファイル拡張子* :プロジェクトのトップレベルフォルダーに対しては編集できません。これはセミコロンで区切られたファイル拡張子のリストです(例、`xml;svg;wml`)。このリストはファイルがプロジェクトに追加される際どのファイルがフォルダーに追加されるかを決定します。例えば、アクティブで関連するファイルがプロジェクトに追加されると、がファイルが追加されるフォルダー決定します。
- *検証* :フォルダー内のXML ファイルの検証において使用されるDTD またはXML スキーマを指定します。
- *変換* : (i) フォルダー内のXML ファイルの変換において使用されるXSLT ファイルと、(ii) フォルダー内のXSLT ファイルが変換を行うXML ファイルを指定します。
- *出力先* :変換の結果生成されるファイルを保存するフォルダーと拡張子を指定します。*Authentic ビューのSPS ファイル* :フォルダー内のXML ファイルをAuthentic ビューにより閲覧、編集するためのSPS ファイルを指定します。

以下の点に注意してください:

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

詳細に関しては、「プロジェクト | プロパティ」[808] コマンドの説明を参照してください。

プロジェクト内のソース管理

プロジェクトではMicrosoft Visual SourceSafe 互換のソース管理システムがサポートされます。この機能の使用方法については、マニュアル内の対応するユーザーリファレンスのセクション[785]を参照ください。

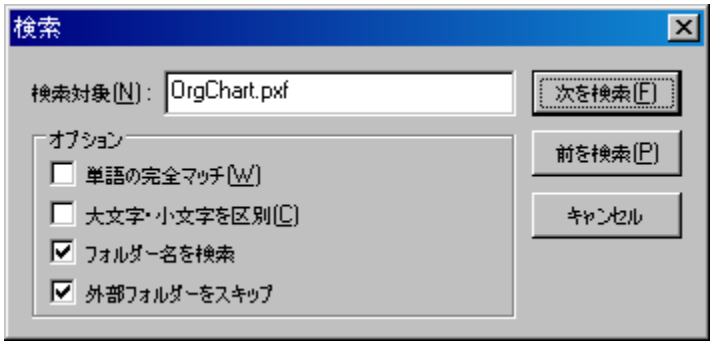
プロジェクトの保存

ファイルの追加や削除、プロジェクトプロパティの修正といったプロジェクトの変更は、「**プロジェクトの保存**」コマンドにより保存されなければなりません。

プロジェクト内の検索

ファイルの名前、またはファイル名の一部からプロジェクト内部にあるファイルやフォルダーを検索することができます。検索が行われると、該当するファイルやフォルダーが1つずつハイライトされた状態で表示されます。

検索を行うには、検索を行うプロジェクトフォルダーをプロジェクトウィンドウにて選択し、「**編集 | 検索**」メニューコマンドを選択するか、「Ctrl」+ F キーを押下します。検索ダイアログが表示されるので、検索を行う文字列を入力し、目的に従うかたちで検索オプションを指定します(以下のスクリーンショットを参照)。検索オプションについては以下を参照ください。



以下の検索オプションが利用できます:

- 単語の完全マッチ: 入力された検索語とファイルまたはフォルダー名が完全に一致するかチェックされ、厳格な検索を行うことができます。ドット (".") の前後に入力された文字列も単語として認識され、例えばファイルの拡張子だけを入力して検索を行うことができます。
- 大文字小文字を区別: 検索語として入力された文字列の大文字と小文字を区別するかどうか指定することができます。
- フォルダー名を検索: フォルダー名を検索の対象に含めます。選択されていない場合、ファイル名だけが検索されます。
- 外部フォルダーをスキップ 800: 外部フォルダーを検索の対象に加える、または対象から外す事ができます。外部フォルダーはシステムまたはネットワーク上に実在しているフォルダーで、プロジェクト内部で論理的に作成されたプロジェクトフォルダーとは異なるものです。

検索が正常に行われた場合、最初にマッチしたアイテムがプロジェクトウィンドウにてハイライトされます。検索ダイアログにある「次を検索」ならびに「前を検索」ボタンをクリックすることで、その他のマッチを参照することができます。

プロジェクトの更新

外部フォルダーに対して変更がなされた場合、プロジェクトが再ロードされるまでその変更はプロジェクトウィンドウに反映されません。

(C) 2015-2021 Altova GmbH

18.2　プロジェクトを使用する

プロジェクトを使うことで同時に複数のファイルに対して設定を行ったり、バッチコマンドを設定し、ワークスペースの管理を効率的に行うことができます。プロジェクトを使用することで、作業のスピードが飛躍的に向上します。

プロジェクトを使う利点

プロジェクトを使用する利点を以下に示します:

- ファイルの拡張子などの条件により、複数のファイルやフォルダーを1つのフォルダーにグループ化することができます。
- スキーマやXSLT ファイルをフォルダーに割り当てることができます。複数のスキーマやファイルを使って XML ファイルの検証を素早く行うことができます。XML ファイルを複数のフォルダーに追加して、各フォルダーにて異なるスキーマやXSLT を定義してください。
- 各フォルダーに対して、バッチ処理を適用することができます。バッチ処理で使用することのできるコマンドは以下に記されます。
- 変換の出力先となるフォルダーを指定することができます。

素早いアクセスを実現するリソース管理

フォルダーやファイルリソースをツリー構造で管理することができ、プロジェクト内にある様々なフォルダーやファイルの全容を確認し、目的のファイルへ素早く辿りつけることができます。プロジェクトウィンドウで表示されているファイルをダブルクリックして開いてください。プロジェクトへのファイルやフォルダーの追加は、必要に応じて素早く行うことができ、必要でないファイルやフォルダーを削除することもできます。他のプロジェクトで作業を行ないたい場合は、現在開かれているプロジェクトを閉じて、目的のプロジェクトを開いてください。

バッチ処理

フォルダー内にあるファイルのバッチ処理は、それがトップレベルのプロジェクトフォルダーであるかどうかに関わらず、**そのフォルダーのコンテキストメニューから利用できます**（フォルダーにて右クリック）。バッチ処理のステップは以下のとおりです:

1. フォルダーのプロパティダイアログにて検証や変換に使用するファイルを定義します。
2. 変換の出力先となるフォルダーを指定します。出力先のフォルダーが指定されなかった場合、プロジェクトツリー上の親フォルダーにて指定されている出力先フォルダーが使用されます。
3. コンテキストメニューからバッチ処理コマンドを使用します。メニューオプションのXML、DTD/スキーマ、または**XSL/XQuery** から対応するコマンドを選択すると、メインウィンドウにてアクティブになっているドキュメントに対してのみコマンドが実行され、プロジェクトウィンドウのプロジェクトフォルダーには影響を与えません。

（トップレベルまたはその他の）プロジェクトフォルダーのコンテキストメニューでは、以下のコマンドがバッチ処理で利用できます:

- *整形式をチェック*：バッチ処理中に発見したエラーは、全てメッセージウィンドウに表示されます。
- *検証*：バッチ処理中に発見したエラーは、全てメッセージウィンドウに表示されます。
- *変換*：フォルダーのプロパティダイアログにて指定した出力フォルダーに変換の出力が保存されます。フォルダーが指定されなかった場合、プロジェクトツリー上の親フォルダーにて指定されている出力先フォルダーが使用されます。親（または祖先）フォルダーにて出力先フォルダーが定義されていない場合、ドキュメントウィンドウが開かれ、各変換の結果が表示されます。XSL-FO 変換ではXML またはFO ドキュメントをPDF へ変換します。
- *DTD/スキーマの生成*：スキーマが生成される前に、出力先フォルダーを指定するよう促されます。生成されたスキーマファイルはそのフォルダーに保存され、個別のウィンドウにて表示されます。

メモ:　バッチ処理を実行するには、プロジェクトウィンドウの対応するフォルダーにてコンテキストメニューを使用してください。メニューオプションのXML、DTD/スキーマ、またはXSL/XQuery から対応するコマンドを使用しないでください。これらのコマンドは、メインウインドウにてアクティブになっているドキュメントに対して実行されます。

RaptorXML Server を使用しての検証とXSLT/XQuery

プロジェクトフォルダーのコンテキストメニューコマンドを使用して、RaptorXML Server をハイパフォーマンスXML 検証とXSLT/XQuery 変換に使用することができます。詳細に関してはRaptorXML Server 683 のセクションを参照してください。

19 RaptorXML Server

Altova RaptorXML Server が、インストールされ、ネットワーク上でライセンス与えられており、XMLSpy のインストールにアクセスが与えられている場合、XML および XBRL* ドキュメントを検証し、XSLT と XQuery の変換[699]. の実行をするために、RaptorXML Server (RaptorXML または Raptor と以下省略されます) を使用することができます。XMLSpy プロジェクト フォルダー内のアクティブなドキュメント または すべてのドキュメントを検証することができます。検証の結果 は、メッセージウィンドウの GUI に表示されます。

XMLSpy を使用して以下を行うことができます (i) ドキュメントの検証 (ii) XMLSpy のエンジンまたは RaptorXML Server を使用して、XSLT/XQuery の変換の実行。Raptor を使用する利点の一つは、広範囲に及ぶ検証の個別の検証を構成できることです。更に、Raptor オプション のセットを XMLSpy 内の"構成"として、保管することができ、特定の Raptor 検証のための構成を選択します。Raptor の使用は、大きなデータコレクションを検証する際にとても役に立ちます。

メモ 実際のパフォーマンスは、検証のために RaptorXML Server により使用される PC プロセッサーコアの数によります。使用するコア数が多いほど、処理は速くなります。

***メモ** Raptor には、2 つのエディションがあります。: (XML 検証のための) *RaptorXML Server* および (XML および XBRL 検証のための) *RaptorXML+XBRL* 。XBRL ドキュメントを検証する場合は、RaptorXML+XBRL Server を使用します。RaptorXML(+XBRL) Server の詳細に関しては、 Altova Web サイト および ユーザーマニュアル: *RaptorXML Server* および *RaptorXML+XBRL Server* を参照してください。

メモ RaptorXML Server は Web ソケットの更新をサポートしないため HTTP プロキシと一緒に使用することはできません。この問題が発生すると RaptorXML Server ホストをプロキシを無視するリストに追加してください。

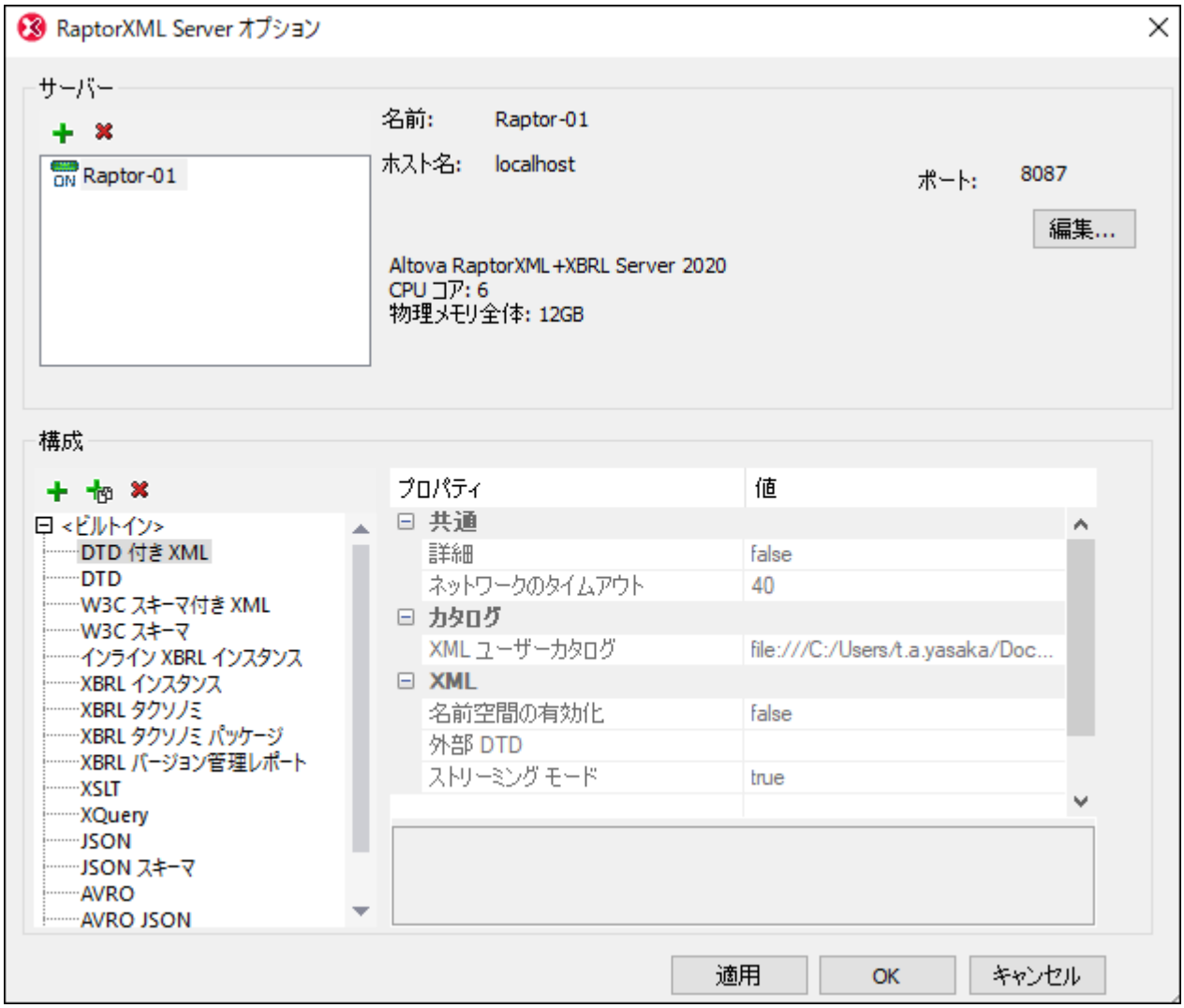
RaptorXML Server を使用して検証または変換する方法

RaptorXML Server を使用して、XML または XBRL ドキュメントを検証、または、XSLT や XQuery 変換を実行するには、XMLSpy は、サーバーへのアクセス方法、検証のため Raptor にパスするオプションなどの使用する RaptorXML Server についての情報を必要とします。XMLSpy 内でこの情報は以下のとおり管理されます:

1. *Raptor サーバーのプールにサーバーを追加する*[684] . このステップでは、, RaptorXML Servers が、プールに追加され、XMLSpy 内に補完されている各サーバーの情報にアクセスすることができます。各サーバーは、名前により識別されています。
2. *各 サーバーのための構成の定義*[685]. 構成は、Raptor 検証 オプションのセットです。各サーバーは、複数の構成を持つことができます。検証に関しては、構成を1つ選択すると、選択された構成がアクティブな構成になります。
3. *検証に使用する サーバー構成 の選択*[688] 。サーバーとその構成をアクティブな 構成に選択します。アクティブな構成は、後のすべての検証で使用されます。
4. *検証*[688] *または Raptorを使用して XSLT/XQuery 変換の実行*[699]。

19.1 サーバーとサーバー構成の追加

RaptorXML Server オプションダイアログ（下のスクリーンショット、「ツール | Raptor Servers の管理」994）内で、Raptor Servers で使用することのできるプールに複数の Raptor Servers の追加684 し、各サーバーのための複数の構成の定義685 に追加することができます。各サーバーのために定義された構成と共にサーバーを追加するには、「ツール | Raptor Servers および構成」998 サブメニューを使用します。このサブメニューで、Raptor 検証のために使用するを選択します。



Raptor Server を追加する

ダイアログの サーバーペイン（下のスクリーンショット）内から、「**サーバーの追加**」アイコンをクリックします。アイコンをクリックして、Raptor Server、がインストールされているマシンの名前(ホスト名)、および Raptor Server のポートを認識するための名前を入力します。「**OK**」をクリックして設定を保存します。

- *名前:* 選択されるすべての文字列。XMLSpy 内では、特定のRaptorXML Server を認識するために使用されています。
- *ホスト名:* Raptor Server がインストールされているマシンのネットワークの名前 またはIP アドレス。処理はホスト名よりもIP アドレスを使用した場合のほうが速くなります。`localhost`（ローカルマシン）に対応するIP アドレスは`127.0.0.1` です。
- *ポート:* Raptor Server がアクセスされるポート。このポートは、（`server_config.xml` という名前の）Raptor の構成ファイル内で指定されています。ポートは、固定され、サービスへのリクエストが対応できるように知らされている必要があります。Raptor 構成ファイルの詳細は、*RaptorXML Server* および *RaptorXML+XBRL Server* のユーザーマニュアルを参照してください。

サーバー情報を入力すると、「OK」をクリックします。入力したサーバー名が（ペインの左側の）サーバーリストに表示されます。サーバーの名前の横に緑のアイコンが表示され、サーバーが開始され、実行中であることを示しています。サーバーの詳細はペインに表示されています（*上のスクリーンショット参照*）。赤いアイコンはサーバーがオフラインであることを示します。サーバーを検出することができない場合、エラーメッセージが表示されます。

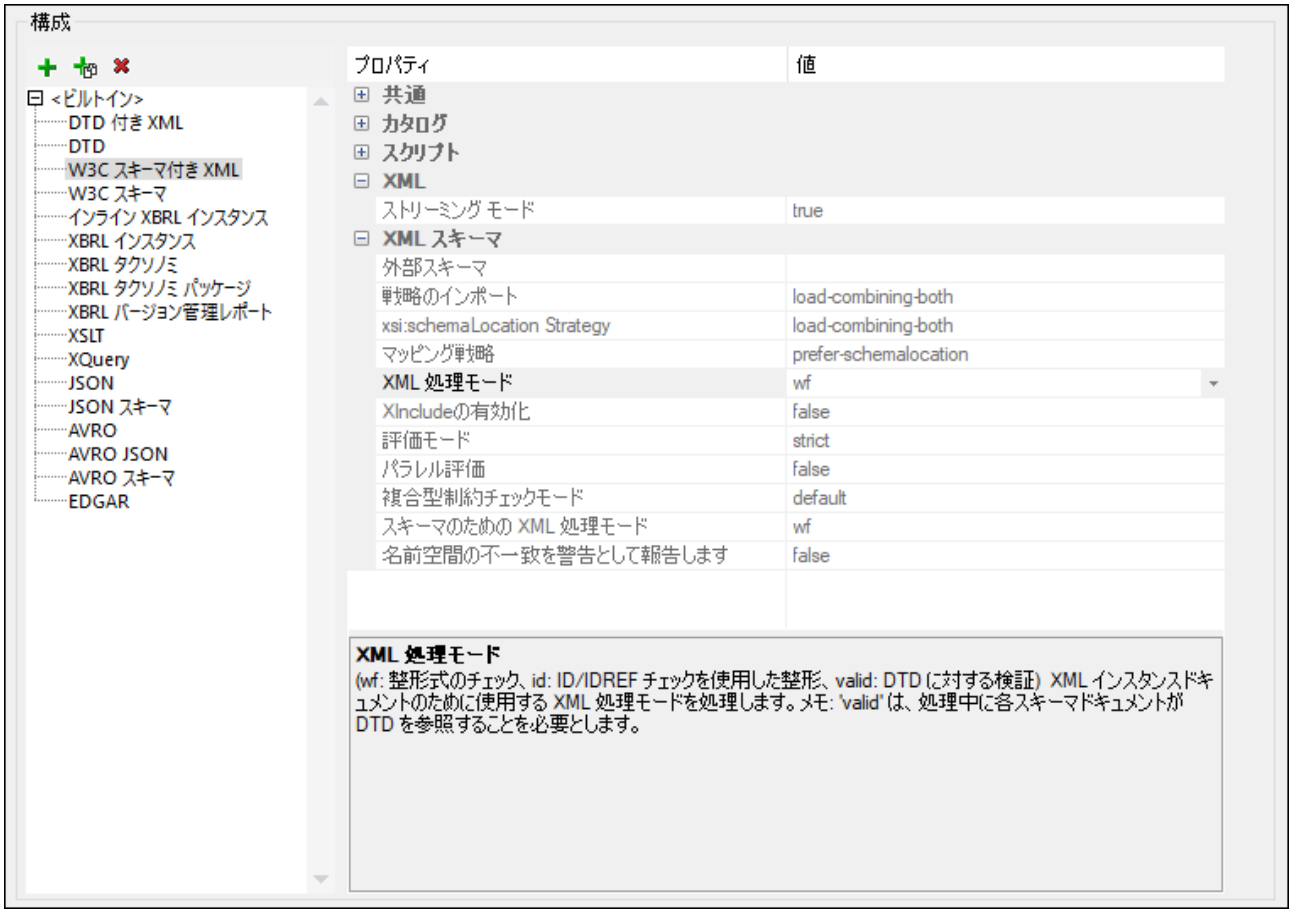
メモ　Raptor Server は、サーバーが追加される際に実行中である必要があります。これはXMLSpy がサーバーの情報を取得し、保管するために必要です。サーバーが追加された後、サーバーがオフラインまたは検出されない場合、これらのシナリオは赤いアイコンまたはエラーメッセージで通知されます。

サーバーの名前、ホスト名、またはポートを編集する場合、左側のペインのサーバーを選択して、「**編集**」ボタンをクリックし、表示されるダイアログ内の情報を変更します。プールからサーバーを削除する場合は、サーバーを選択して「**選択されたサーバーを削除**」アイコンをクリックします。

Server 構成

構成とは、RaptorXML 検証オプションのセットです。サーバーが追加されると、構成を持ち、`default` と名前が付けられます。これはRaptorXML オプションのセットに設定されたデフォルトの値です。他のオプションの値を含む新しい構成を追加することもできます。複数のサーバー構成を追加した後、アクティブな構成になる1 つの構成を選択します。これが「**サーバー上で検証**」コマンドが実行される時に使用される構成です。

構成ペインは2 つの部分により構成されています: (i) それぞれが変更可能なドキュメント型のリストを含む構成を表示する左側のペイン、(ii) 左側のペインで選択されたドキュメント型のための検証オプションを表示する右側のペイン。右側のペインの下にはお選択されたオプションの説明が表示されています（*上のスクリーンショット参照*）。



構成の追加

RaptorXML Server オプションダイアログの構成ペイン内から(*上のスクリーンショット*)「**構成の追加**」をクリックします。デフォルトのオプション値に新規構成が追加されます。「**選択された構成をコピー**」をクリックすることにより新規構成を作成することもできます。これによりコピーされた構成のオプション値と同じ値を持つ新しい構成が作成されます。新たしい構成は、型 `config<X>` のデフォルトの名前と共に作成されます。構成の名前をダブルクリックして新しい名前を入力して編集します。構成のオプション値のすべてを編集することができます。

構成のオプション値の編集

左側のペインからドキュメント型を選択します。これにより右側のペイン内に選択されたドキュメント型の検証オプションが表示されます。オプションの値を編集するには、(オプションの値の型に従い)以下を行います:

- 値が定義済みの値のセットの1つの場合、そのオプションの値の列のコンボボックスから値を選択します。
- 値に制約がある場合、オプションの値フィールドをクリックして、値を入力します。
- 値がファイルパスの場合、値を入力できるだけでなく、オプションの値の列の「**参照**」ボタンを押して、使用するファイルを参照します。

オプションを選択すると、右側のペインの下のボックスに詳細が表示されます。各オプションの詳細は、RaptorXML Server および RaptorXML(+XBRL) Server ユーザーマニュアルのコマンドラインインターフェイスのセクションを参照してください。

構成の削除

左側のペインから、削除する構成を選択して、「**選択された構成を削除**」をクリックします。

Visual Studio および Eclipse 内の XMLSpy

Visual Studio[729] および Eclipse[734], に XMLSpy が統合されると、IDE のアクティブの構成が XMLSpy のスタンドアロンバージョン内のアクティブな構成として設定されます。

19.2 RaptorXML Server を使用して検証

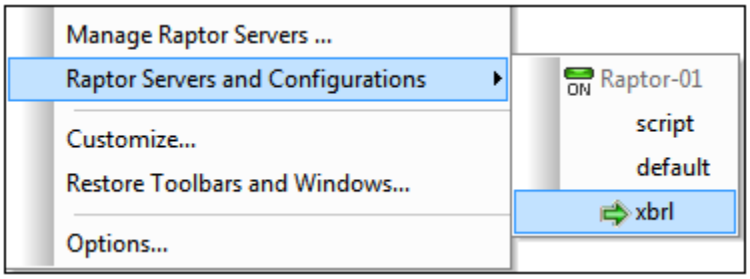
RaptorXML Server を使用して、XML および XBRL* ドキュメント を検証することができます。検証には 2 つのステップが含まれます:

- 検証のために使用されるサーバーとサーバー構成を選択します。
- (**サーバー上での検証** コマンドを使用した *以下を参照*) 検証の実行

***メモ** Raptor には、2 つのエディションがあります。: (XML 検証のための) *RaptorXML Server* および (XML および XBRL 検証のための) *RaptorXML+XBRL* 。XBRL ドキュメントを検証する場合は、RaptorXML+XBRL Server を使用します。RaptorXML(+XBRL) Server の詳細に関しては、 Altova Web サイト および ユーザーマニュアル: *RaptorXML Server* および *RaptorXML+XBRL Server* を参照してください。

使用するサーバー構成の選択

複数のサーバーで複数の構成を定義した場合、サーバーとそのアクティブな構成として 1 つの構成を選択することができます。アクティブな構成は、以降の検証に使用されます。カーソルを「**ツール | Raptor Servers および 構成**」コマンドにポイントして、追加されたすべてのサーバーとその詳細が含まれるサブメニューから、を含む表示されるサブメニューが表示されます (*下のスクリーンショット参照*)。アクティブな構成と設定するサーバー構成を選択します。したのスクリーンショットでは、`Raptor-01` という名前のサーバーの (緑色の矢印で示されている) `xbrl` 構成がアクティブな構成として選択されています。



RaptorXML Server を使用して検証する

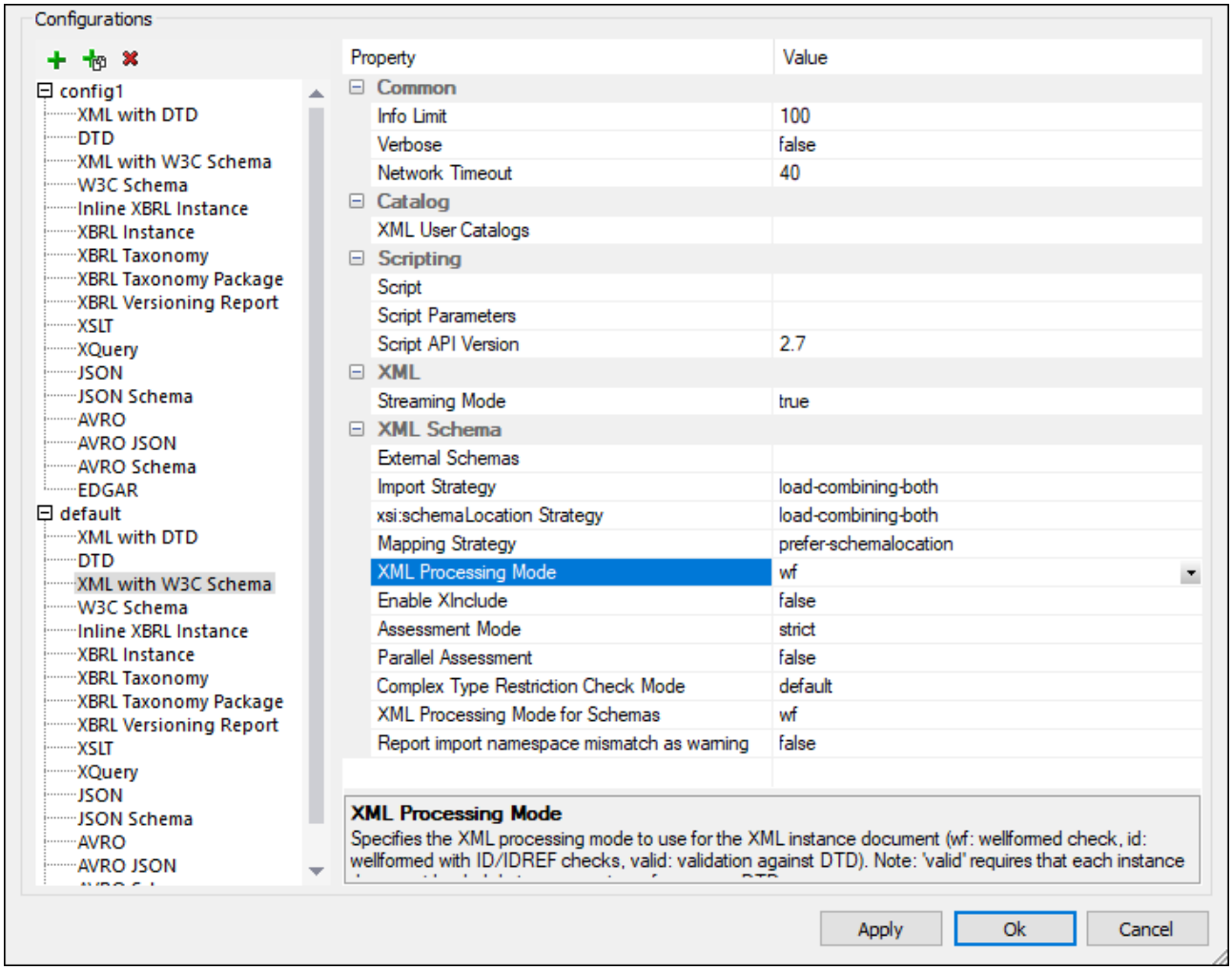
XMLSpy の検証エンジンまたは、RaptorXML Server を使用して XML および XBRL ドキュメントを検証することができます。以下の方法を使用してください:

- ツールバーアイコン **サーバー上で検証する**をクリックします。
- コマンド「**XML | サーバー上で XML の検証する**」(*ハイパフォーマンス*)(Ctrl+F8) を選択します。
- プロジェク、フォルダーまたはファイル内の入力ヘルパーのプロジェクトを右クリックし、「**サーバー上の XML を検証する**」(*ハイパフォーマンス*) を選択して、選択されたオブジェクト内の XML または XBRL データを検証します。

メモ テキストビュー、スキーマビュー、、および グリッドビュー内で Raptor 検証を使用することができます。

19.3 検証オプション

このセクションは、検証されるドキュメントの型により整理されています（下のスクリーンショットの左側のペインを参照してください）。例えば、*W 3C スキーマを持つ XML は*、XML ドキュメントを W 3C XML スキーマに対して検証します。左側のペインで検証が選択されると、検証の型で使用することのできる RaptorXML Server 検証 オプションが右側のペインに表示されます。これらのオプションは、*スクリプト*および *XML スキーマ*などのグループ別に整理されています（*下のスクリーンショット参照*）。



このセクションのサブセクションは、それぞれの RaptorXML Server 検証 オプションの説明へのリンクを含みます。

19.3.1 共通のオプション

オプションはすべての型の検証で共通です。

共通

[情報の制限](#)
[詳細](#)

ネットワークタイムアウト

カタログ

XML ユーザーカタログ

19.3.2 DTD に対して XML

DTD に対して XML データを検証するためのオプション

共通

情報の制限
詳細
ネットワークのタイムアウト

カタログ

XML ユーザーカタログ

XML

名前空間を有効化する
外部 DTD
ストリーミングモード

19.3.3 DTD

DTD の検証のためのオプション

共通

情報の制限
詳細
ネットワークタイムアウト

カタログ

XML ユーザーカタログ

スクリプト

スクリプト

(C) 2015-2021 Altova GmbH

スクリプト パラメーター
スクリプト API バージョン

19.3.4 W3C スキーマに対して XML

XML スキーマに対して XML データを検証するためのオプション

共通

情報の制限
詳細
ネットワークのタイムアウト

カタログ

XML ユーザーカタログ

スクリプト

スクリプト
スクリプト パラメーター
スクリプト API バージョン

XML

ストリーミングモード

XML スキーマ

外部 スキーマ(xsd)
インポートストラテジー(schema-imports)
xsi:schemaLocation Strategy (schemalocation-hints)
ストラテジーのマッピング(schema-mapping)
XML 検証モード(xml-mode)
XInclude の有効化 (xinclude)
評価モード
パラレル評価

19.3.5 W3C スキーマ

XML スキーマの検証のためのオプション

共通

情報の制限

詳細
ネットワークのタイムアウト

カタログ

XML ユーザーカタログ

スクリプト

スクリプト
スクリプト パラメーター
スクリプト API バージョン

XML スキーマ

インポートストラテジー(schema-imports)
xsi:schemaLocation Strategy (schemalocation-hints)
ストラテジーのマッピング(schema-mapping)
XML 検証モード(xml-mode)
XInclude の有効化(xinclude)

19.3.6 XBRL インスタンス

XBRL インスタンドドキュメントの検証のためのオプション

共通

情報の制限
詳細
ネットワークのタイムアウト

カタログ

XML ユーザーカタログ

スクリプト

スクリプト
スクリプト パラメーター
スクリプト API バージョン

XML スキーマ

インポートストラテジー(schema-imports)
xsi:schemaLocation Strategy (schemalocation-hints)
ストラテジーのマッピング(schema-mapping)

XInclude の有効化 (xinclude)
パラレル評価

XBRL

ディメンション拡張子の有効化 (dimensions)
拡張可能な列挙 拡張子の有効化 (extensible-enumerations)
ユニットレジストリ名前空間の有効化 (utr)
XBRL スペックスキーマのプリロード (preload-xbrl-schemas)
タクソノミ パッケージ
参照されているDTS のみを検証(validate-dts-only)
XBRL 矛盾をエラーとして扱う(treat-inconsistencies-as-errors)

XBRL フォーミュラ

フォーミュラ拡張子の有効化 (formula)
アサーション重要度の拡張の有効化 (assertion-severity)
フォーミュラスペックスキーマのプリロード (preload-formula-schemas)
条件を満たさないアサーションのレポート
検証メッセージ言語 (message-lang)
検証メッセージの役割 (message-role)

XBRL テーブル

テーブル拡張子の有効化 (table)
テーブルスペックスキーマのプリロード (preload-table-schemas)
テーブルリンクベース名前空間

19.3.7 XBRL タクソノミ

XBRL タクソノミの検証のためのオプション

共通

情報の制限
詳細
ネットワークのタイムアウト

カタログ

XML ユーザーカタログ

スクリプト

スクリプト
スクリプト パラメーター
スクリプト API バージョン

- XML スキーマ

  インポートストラテジー(schema-imports)
  xsi:schemaLocation Strategy (schemalocation-hints)
  ストラテジーのマッピング(schema-mapping)
  XInclude の有効化(xinclude)

- XBRL

  ディメンション拡張子の有効化(dimensions)
  Enable 拡張可能な列挙 拡張子(extensible-enumerations)
  XBRL スペックスキーマのプリロード(preload-xbrl-schemas)
  タクソノミパッケージ
  XBRL 矛盾をエラーとして扱う(treat-inconsistencies-as-errors)

- XBRL フォーミュラ

  フォーミュラ拡張子の有効化(formula)
  フォーミュラスペックスキーマのプリロード(preload-formula-schemas)

- XBRL Table

  テーブル拡張子の有効化(table)
  テーブルスペックスキーマのプリロード(preload-table-schemas)
  テーブルリンクベース名前空間

19.3.8 XSLT

XSLT ドキュメントの検証のためのオプション

- 共通

  情報の制限
  詳細
  ネットワークタイムアウト

- カタログ

  XML ユーザーカタログ

- XML スキーマ

  インポートストラテジー(schema-imports)
  xsi:schemaLocation ストラテジー(schemalocation-hints)
  マッピングストラテジー(schema-mapping)
  XML 検証モード(xml-mode)
  XInclude の有効化(xinclude)

(C) 2015-2021 Altova GmbH

- Java 拡張子

  Java 拡張子の無効化 (javaext-disable)
  バーコード拡張子のロケーション (javaext-barcode-location)

- チャート拡張子

  チャート拡張子の無効化(chartext-disable)

- .NET 拡張子

  .NET 拡張子の無効化 (dotnetext-disable)

- XEngines 共通

  XML をPSVI と共にロード(load-xml-with-psvi)

- XSLT

  XSLT エンジン バージョン (xslt-バージョン)
  テンプレートモード
  テンプレートエントリポイント

19.3.9 XQuery

XQuery ドキュメントの検証のためのオプション

- 共通

  情報の制限
  詳細
  ネットワークタイムアウト

- カタログ

  XML ユーザーカタログ

- XML スキーマ

  インポートストラテジー(schema-imports)
  xsi:schemaLocation ストラテジー(schemalocation-hints)
  マッピングストラテジー(schema-mapping)
  XML 検証モード (xml-mode)
  XInclude の有効化 (xinclude)

- Java 拡張子

  Java 拡張子の無効化 (javaext-disable)
  バーコード拡張子のロケーション (javaext-barcode-location)

- チャート拡張子

  チャート拡張子の無効化(chartext-disable)

- .NET 拡張子

  .NET 拡張子の無効化 (dotnetext-disable)

- XEngines 共通

  XML をPSVI と共にロード(load-xml-with-psvi)

- XQuery

  XQuery エンジン バージョン (xquery-version)
  XML 宣言の削除

19.3.10 JSON

JSON (インスタンス) ドキュメントの検証のためのオプション

- 共通

  情報の制限
  詳細
  ネットワークタイムアウト

- カタログ

  XML ユーザーカタログ

- JSON 検証

  フォーマットチェックの無効化
  JSON ライン
  コメント付きのJSON

19.3.11 JSON スキーマ

JSON スキーマドキュメントの検証のためのオプション

共通

情報の制限
詳細
ネットワークタイムアウト

カタログ

XML ユーザーカタログ

JSON 検証

フォーマットチェックの無効化

19.3.12 EDGAR

EDGAR (エドガー、Electronic Data Gathering, Analysis, and Retrieval) は、米国証券取引委員会SEC (Securities and Exchange Commission) に企業により提出されたファイルを自動的に収集、検証、および分類するシステムです。EDGAR を介して検証する場合、Raptor は、内部 EDGAR スクリプトを使用して XBRL インスタンスドキュメントを検証します。以下の追加オプションの設定することことができます。

EDGAR スクリプト パラメーター

EDGAR スクリプトは、EDGAR Filing Manual Volume II: EDGAR Filingで説明されているとおり、追加のチェックを行います。スクリプトは、以下のスクリプト パラメーターを指定することを許可します:

CIK	登録者名のCIK
submissionType	EDGAR 送信の型、例えば '10-K'
cikList	カンマで区切られたCIK のリスト: ','
cikNameList	'|Edgar|' により区切られる、cikList 内の各 CIK のための公式の登録者名のリスト
forceUtrValidation	UTR 検証を強制的に有効化するためにtrue に設定
edbody-url	埋め込まれたHTML フラグメントの検証に使用されるedbody.dtd へのパス
edgar-taxonomies-url	カンパニー拡張タクソノミタクソノミからの参照を許可するファイルのリストを含む edgartaxonomies.xml へのパス

共通

情報の制限
詳細
ネットワークタイムアウト

カタログ

XML ユーザーカタログ

XML スキーマ

インポートストラテジー(schema-imports)
xsi:schemaLocation ストラテジー(schemalocation-hints)
マッピングストラテジー(schema-mapping)
XInclude の有効化 (xinclude)
パラレル評価

XBRL

ディメンション拡張子の有効化 (dimensions)
拡張可能な列挙 拡張子の有効化 (extensible-enumerations)
XBRL スペックスキーマのプリロード (preload-xbrl-schemas)
タクソノミ パッケージ
XBRL 非整合性をエラーとして扱う(treat-inconsistencies-as-エラー)

XBRL フォーミュラ

フォーミュラ 拡張子の有効化(formula)
フォーミュラ スペックスキーマのプリロード (preload-formula-schemas)

XBRL テーブル

テーブル拡張子の有効化 (table)
テーブルスペックスキーマのプリロード(preload-table-schemas)
テーブルリンクベース名前空間

(C) 2015-2021 Altova GmbH

19.4 RaptorXML Server使用しての XSLT と XQuery

RaptorXML Server を使用して以下を行うことができます (i) XSLT 変換 (ii) XML ドキュメント上のXQuery アップデートの実行。これらのアクションはプロジェクト 676 のみで使用することができ、以下の3 つのステップを含みます:

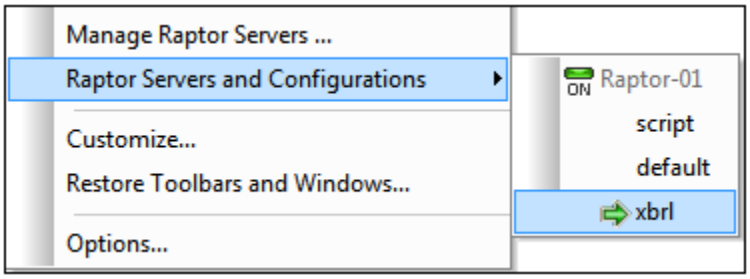
- ジョブに使用されるサーバーとサーバーの構成の選択。
- プロジェクトフォルダー 677 の設定と(プロジェクトプロパティダイアログ内で 677)使用するXSLT/XQuery ファイルの指定。XSLT/XQuery ファイルは、プロジェクトフォルダー内のすべてのXML ファイルのXSLT とXQuery 変換に使用されたファイルであるフォルダーのプロジェクトプロパティダイアログ 677 内に割り当てられます。プロジェクトフォルダー内の個々のXML ファイルにXSLT/XQuery ファイルを割り当てることはできません。XSLT/XQuery ファイルはフォルダー全体のみに割り当てることができます。
- XSLT 変換またはXQuery 更新および実行

メモ XSLT またはXQuery ドキュメントがJava 拡張関数 または.NET 拡張関数を使用する場合、ファイルパスを使用して、JAR ファイル(Java) または外部 (登録されていない) アセンブリファイル(.NET)を検索します。これは、XMLSpy および RaptorXML Serverを使用して、同じ XSLT/XQuery ドキュメントが変換および実行に使用される場合、JAR ファイルへのまたはアセンブリファイルへのファイルパスがこれらのファイルを正確に検索する必要があります。

メモ RaptorXML Server がXMLSpy と同じマシン上にある場合、パフォーマンス向上のために、サーバー設定を `true` の値を持つ **`server.unrestricted-filesystem-access`** と設定します。更に詳しい情報は、RaptorXML Server 構成ファイルのドキュメントを参照してください。

使用するサーバー構成の選択

複数のサーバーで複数の構成を定義した場合、サーバーとそのアクティブな構成として1 つの構成を選択することができます。アクティブな構成は、以降の検証に使用されます。カーソルを「**ツール | Raptor Servers および構成**」コマンドにポイントして、追加されたすべてのサーバーとその詳細が含まれるサブメニューから、を含む表示されるサブメニューが表示されます (*下のスクリーンショット参照*)。アクティブな構成と設定するサーバー構成を選択します。したのスクリーンショットでは、`Raptor-01` という名前のサーバーの(緑色の矢印で示されている) `xbrl` 構成がアクティブな構成として選択されています。



XSLT 変換の実行

XMLSpy またはRaptorXML Server のXSLT エンジンを使用してXSLT 変換 の実行をするには以下を行います:

- 変換されるXML ファイルが存在するプロジェクトフォルダーを右クリックします。このフォルダーはプロジェクトフォルダー全体またはプロジェクトの階層内の個別のフォルダーであることができます。
- 表示されるメニューから、「**サーバー上でのXSL 変換 (ハイパフォーマンス)**」を選択します。

メモ プロジェクトフォルダー内の個々のXML ファイルにXSLT/XQuery ファイルを割り当てることはできません。XSLT/XQuery ファイルはフォルダー全体のみに割り当てることができます。セクションの始め 699 を参照してください。

詳細に関しては、XSLT 320 およびXSLT 変換 888 のセクションを参照してください。

XQuery の更新/実行の実行

XMLSpy またはRaptorXML Server のXSLT エンジン を使用してXQuery 更新/変換 の実行をするには以下を行います:

- 更新および実行するXQuery またはXML ファイル が存在するプロジェクト フォルダーを右クリックします。このフォルダーはプロジェクト フォルダー全体またはプロジェクト の階層内の個別のフォルダーであることができます。
- 表示されるメニューから、「**サーバー上でのXQuery 更新/実行 (ハイパフォーマンス)**」を選択します。

メモ プロジェクト フォルダー内の個々のXML ファイルにXSLT/XQuery ファイルを割り当てることはできません。XSLT/XQuery ファイルはフォルダー全体のみに割り当てることができます。セクションの始め 699 を参照してください。

詳細に関しては、XQuery 332 および XQuery/Update 実行 893 のセクションを参照してください。

20 ファイル/ディレクトリの比較

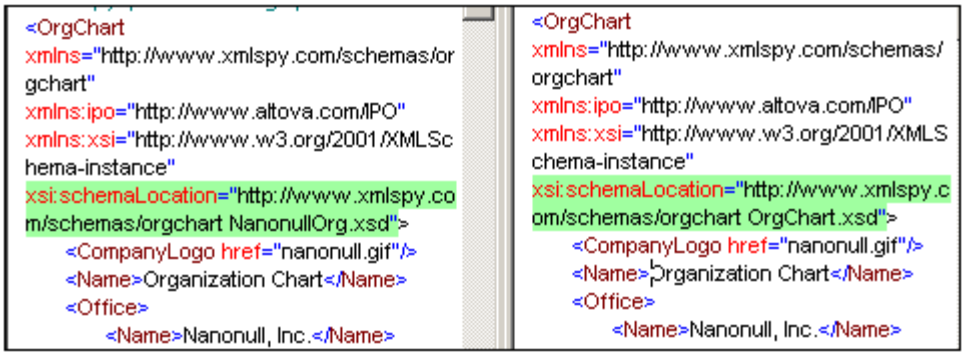
XMLSpy にはファイル比較ならびにディレクトリ比較機能が搭載されています。ファイル比較とディレクトリ比較は、「**ツール**」メニューの「**開いているファイルと比較**」ならびに「**ディレクトリを比較**」コマンドからそれぞれアクセスできます。ファイルの比較オプションは、メニューオプションの「**ツール | 比較オプション**」コマンドによりアクセスすることのできる設定ダイアログにて設定することができます。

各コマンドの詳細は*ユーザーリファレンスのセクション*[983]にて記述されます。このセクションのサブセクションでは*ファイル比較*[702]と*ディレクトリ比較*[703]メカニズムの概要について説明します。

20.1 ファイルの比較

ファイル比較機能[983]により、ファイルを開くダイアログやグローバルリソース[660]から選択されたファイルと、アクティブなファイルとの比較を行うことができます。以下ではそのメカニズムの概要を説明します。詳細についてはユーザーリファレンスのセクション[983]を参照ください。

- ファイル比較セッションが開始されると、比較オプション[990]ダイアログにて指定された設定が、そのセッションにて有効になります。
- ファイルはXML ファイル(ドキュメントの構造も比較の対象になります)、またはテキストファイルとして比較することができます。設定ダイアログ[990]にて、(i) グリッドビューまたはテキストビューを選択し、テキストの比較だけを行うのチェックを外すことでXML 比較を、(ii) テキストビューを選択し、テキストの比較だけを行うにチェックを入れることでテキストの比較を行うことができます。
- 選択されたビュー(グリッドビューまたはテキストビュー)にて2つのファイルが隣接して開かれ、両方のファイル間の差異がハイライトされます(以下のスクリーンショットを参照)。

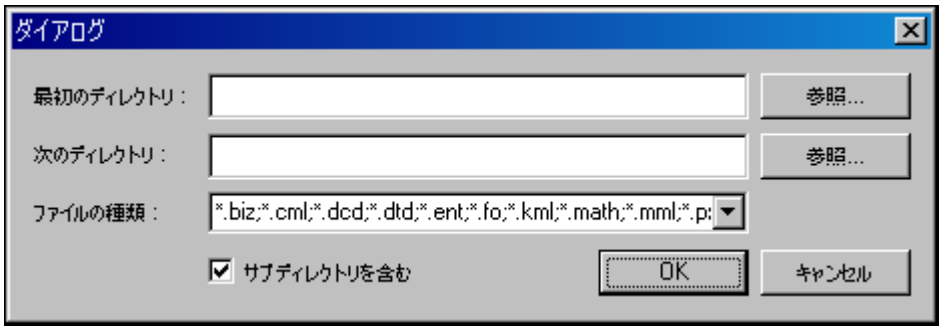


ファイルの比較コントロールウィンドウが表示され、差異のナビゲーションやマージを行うことができます。

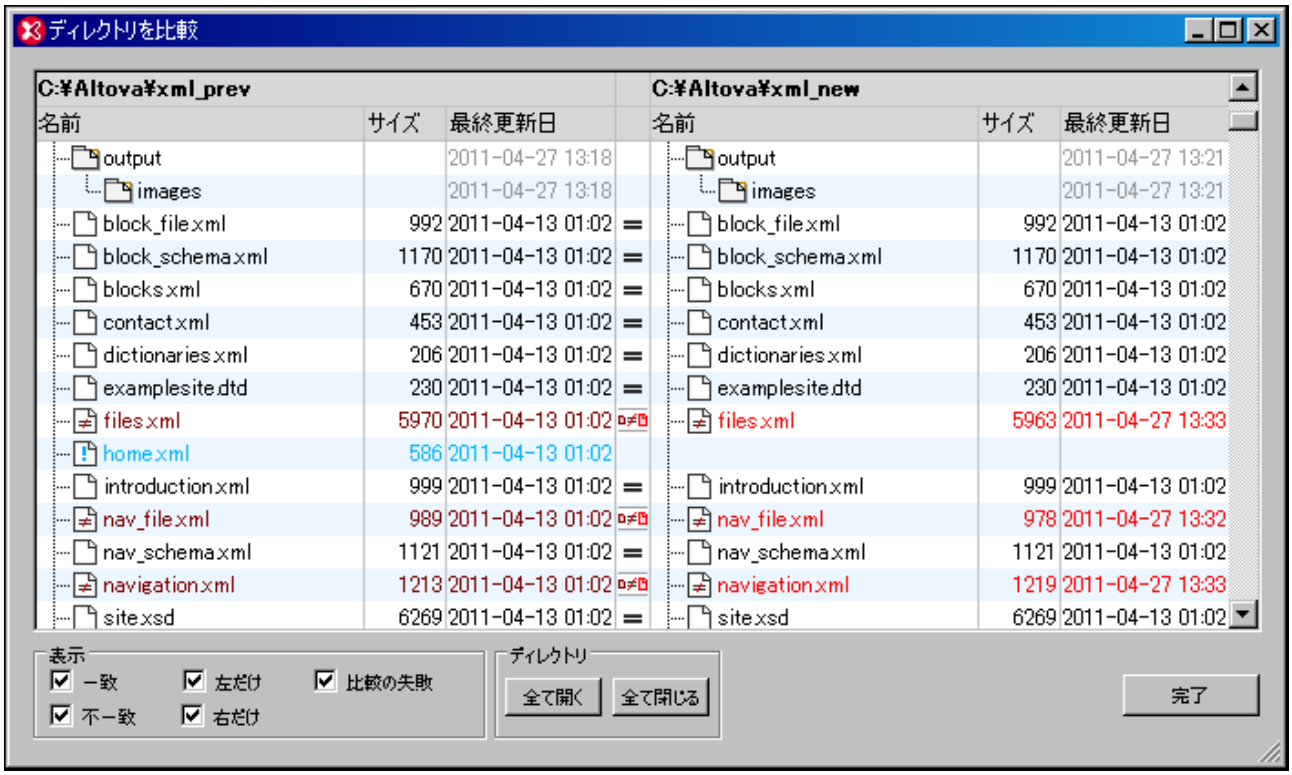
設定ダイアログ[990]では、どのXML コンポーネントを比較の対象とし、何を無視するか設定することができます。詳細については、ユーザーリファレンス[983]の比較オプション[990]セクションを参照ください。

20.2　ディレクトリの比較

ディレクトリの比較機能[987]を使うことで、フォルダーダイアログにて指定したディレクトリ同士を比較することができます。サブディレクトリも比較の対象とするか選択することができ、比較の対象となるファイルの種類を指定することができます。



ディレクトリの比較では、ファイルが欠けているか、同名のファイルが同じものであるかが比較されます。ファイル同士の比較は設定ダイアログ[990]にて指定された設定をベースに行われます。ディレクトリ比較の結果は別のウィンドウにて表示されます（以下のスクリーンショットを参照）。



ディレクトリを比較ウィンドウ内のシンボルの意味や、管理の方法については、ユーザーリファレンス[983]の「ディレクトリを比較」コマンドに関する記述を参照ください。ファイルの行をダブルクリックすると、ファイルの比較を開始することができます。

21 Source Control

The source control support in XMLSpy is available through the Microsoft Source Control Plug-in API (formerly known as the MSSCCI API), versions 1.1, 1.2 and 1.3. This enables you to run source control commands such as "Check in" or "Check out" directly from XMLSpy to virtually any source control system that lets native or third-party clients connect to it through the Microsoft Source Control Plug-in API.

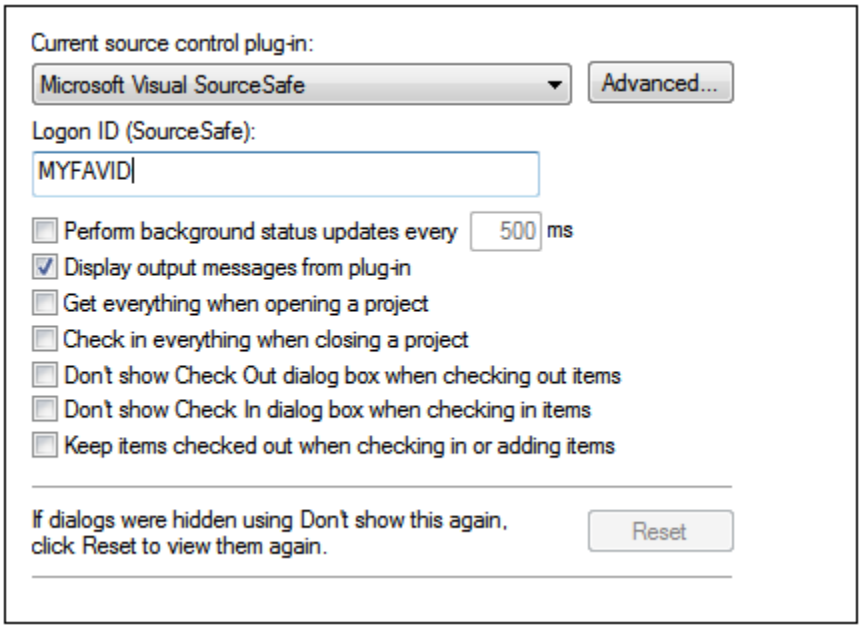
You can use as your source control provider any commercial or non-commercial plug-in that supports the Microsoft Source Control Plug-in API, and can connect to a compatible version control system. For the list of source control systems and plug-ins tested by Altova, see Supported Source Control Systems 707.

Installing and configuring the source control provider

To view the source control providers available on your system, do the following:

1. On the **Tools** menu, click **Options**.
2. Click the **Source Control** tab.

Any source control plug-ins compatible with the Microsoft Source Code Control Plug-in API are displayed in the **Current source control plug-in** drop-down list.



If a compatible plug-in cannot be found on your system, the following message is displayed:

```
"Registration of installed source control providers could not be found or is incomplete."
```

Some source control systems might not install the source control plug-in automatically, in which case you will need to install it separately. For further instructions, refer to the documentation of the respective source control system. A plug-in (provider) compatible with the Microsoft Source Code Control Plug-in API is expected to be registered under the following registry entry on your operating system:

```
HKEY_LOCAL_MACHINE\SOFTWARE\SourceCodeControlProvider\InstalledSCCProviders
```

 (C) 2015-2021 Altova GmbH

Upon correct installation, the plug-in becomes available automatically in the list of plug-ins available to XMLSpy.

Accessing the source control commands

The commands related to source control are available in the **Project | Source Control** menu.

Resource / Speed issues

Very large source control databases might be introducing a speed/resource penalty when automatically performing background status updates.

You might be able to speed up your system by disabling (or increasing the interval of) the **Perform background status updates every ... seconds** option in the **Source Control** tab accessed through **Tools | Options**.

Note: The **64-bit** version of your Altova application automatically supports any of the supported 32-bit source control programs listed in this documentation. When using a 64-bit Altova application with a 32-bit source control program, the **Perform background status updates every ... seconds** option is automatically grayed-out and cannot be selected.

Differencing with Altova DiffDog

You can configure many source control systems (including Git and TortoiseSVN) so that they use Altova DiffDog as their differencing tool. For more information about DiffDog, see https://www.altova.com/diffdog. For DiffDog documentation, see https://www.altova.com/documentation.html.

21.1 Setting Up Source Control

The mechanism for setting up source control and placing files in a XMLSpy project under source control is as follows:

1. If this hasn't been done already, install the source control system (see Supported Source Control Systems[707]) and set up the source control database (repository) to which you wish to save your work.
2. Create a local workspace folder that will contain the working files that you wish to place under source control. The folder that contains all your workspace folders and files is called the local folder, and the path to the local folder is referred to as the local path. This local folder will be bound to a particular folder in the repository.
3. In your Altova application, create an application project folder to which you must add the files you wish to place under source control. This organization of files in an application project is abstract. The files in a project reference physical files saved locally, preferably in one folder (with sub-folders if required) for each project.
4. In the source control system's database (also referred to as source control or repository), a folder is created that is bound to the local folder. This folder (called the bound folder) will replicate the structure of the local folder so that all files to be placed under source control are correctly located hierarchically within the bound folder. The bound folder is usually created when you add a file or an application project to source control for the first time. See the section, Application Project[710], for information about the repository's folder structure.
5. Project files are added to source control using the command **Project | Source Control | Add to Source Control**. When you add a project or a file in a project for the first time to source control, the correct bindings and folder structure will be created in the repository.
6. Source control actions, such as the checking in and out of files, and the removing of files from source control, can be carried out via commands in the **Project | Source Control** submenu. These commands are described in the Project menu subsection[785] of the User Reference.

Note: If you wish to change the current source control provider, this can be done in one of two ways: (i) via the Source Control options (**Tools | Options | Source Control**[1040]), or (ii) in the Change Source Control dialog (**Project | Source Control | Change Source Control**).

21.2 Supported Source Control Systems

The list below shows the Source Control Servers (SCSs) supported by XMLSpy, together with their respective Source Control Clients (SCCs). The list is organized alphabetically by SCS. Note the following:

- Altova has implemented the Microsoft Source Control Plug-in API (versions 1.1, 1.2, and 1.3) in XMLSpy, and has tested support for the listed drivers and revision control systems. It is expected that XMLSpy will continue to support these products if, and when, they are updated.
- Source Code Control clients not listed below, but which implement the Microsoft Source Control Plug-in API, should also work with XMLSpy.

Source Control System	Source Code Control Clients
AccuRev 4.7.0 Windows	AccuBridge for Microsoft SCC 2008.2
Bazaar 1.9 Windows	Aigenta Unified SCC 1.0.6
Borland StarTeam 2008	Borland StarTeam Cross-Platform Client 2008 R2
Codice Software Plastic SCM Professional 2.7.127.10 (Server)	Codice Software Plastic SCM Professional 2.7.127.10 (SCC Plugin)
Collabnet Subversion 1.5.4	• Aigenta Unified SCC 1.0.6 • PushOK SVN SCC 1.5.1.1 • PushOK SVN SCC x64 version 1.6.3.1 • TamTam SVN SCC 1.2.24
ComponentSoftware CS-RCS (PRO) 5.1	ComponentSoftware CS-RCS (PRO) 5.1
Dynamsoft SourceAnywhere for VSS 5.3.2 Standard/Professional Server	Dynamsoft SourceAnywhere for VSS 5.3.2 Client
Dynamsoft SourceAnywhere Hosted	Dynamsoft SourceAnywhere Hosted Client (22252)
Dynamsoft SourceAnywhere Standalone 2.2 Server	Dynamsoft SourceAnywhere Standalone 2.2 Client
Git	PushOK GIT SCC plug-in (see Source Control with Git 724)
IBM Rational ClearCase 7.0.1 (LT)	IBM Rational ClearCase 7.0.1 (LT)
March-Hare CVSNT 2.5 (2.5.03.2382)	Aigenta Unified SCC 1.0.6
March-Hare CVS Suite 2008	• Jalindi Igloo 1.0.3 • March-Hare CVS Suite Client 2008 (3321) • PushOK CVS SCC NT 2.1.2.5 • PushOK CVS SCC x64 version 2.2.0.4 • TamTam CVS SCC 1.2.40
Mercurial 1.0.2 for Windows	Sergey Antonov HgSCC 1.0.1
Microsoft SourceSafe 2005 with CTP	Microsoft SourceSafe 2005 with CTP

Source Control System	Source Code Control Clients
Microsoft Visual Studio Team System 2008/2010 Team Foundation Server	Microsoft Team Foundation Server 2008/2010 MSSCCI Provider
Perforce 2008 P4S 2008.1	Perforce P4V 2008.1
PureCM Server 2008/3a	PureCM Client 2008/3a
QSC Team Coherence Server 7.2.1.35	QSC Team Coherence Client 7.2.1.35
Reliable Software Code Co-Op 5.1a	Reliable Software Code Co-Op 5.1a
Seapine Surround SCM Client/Server for Windows 2009.0.0	Seapine Surround SCM Client 2009.0.0
Serena Dimensions Express/CM 10.1.3 for Win32 Server	Serena Dimensions 10.1.3 for Win32 Client
Softimage Alienbrain Server 8.1.0.7300	Softimage Alienbrain Essentials/Advanced Client 8.1.0.7300
SourceGear Fortress 1.1.4 Server	SourceGear Fortress 1.1.4 Client
SourceGear SourceOffsite Server 4.2.0	SourceGear SourceOffsite Client 4.2.0 (Windows)
SourceGear Vault 4.1.4 Server	SourceGear Vault 4.1.4 Client
VisualSVN Server 1.6	• Aigenta Unified SCC 1.0.6 • PushOK SVN SCC 1.5.1.1 • PushOK SVN SCC x64 version 1.6.3.1 • TamTam SVN SCC 1.2.24

(C) 2015-2021 Altova GmbH

21.3 Local Workspace Folder

The files you will be working with should be saved in a hierarchy inside a local workspace folder (*see diagram below*).

```
Local Workspace Folder
|
|-- MyProject.spp
|-- QuickStart
|   |-- QuickStart.css
|   |-- QuickStart.xml
|   |-- QuickStart.xsd
|-- Grouping
|   |-- Persons
|   |   |-- Persons.xml
```

The application project file (`.spp` file) typically will be located directly inside the local workspace folder (*see diagram above*).

When one or more files in this (workspace) folder are placed under source control, the local workspace folder's structure is partly or wholly reproduced in the repository. For example, if the file `Persons.xml` from the local folder shown above is placed under source control, then the path to it in the repository will be:

```
[RepositoryFolder]/MyProject/Grouping/Persons/Persons.xml
```

The `MyProject` folder in the repository folder is bound to the local folder. Typically it would be the name of the project, but you could give it any name.

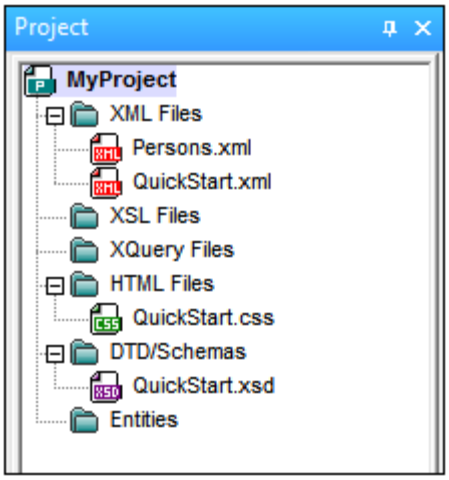
If the entire application project is placed under source control (by selecting the project name in the Projects window and placing it under source control), then the entire local folder structure is recreated in the repository.

Note: Files from outside the local workspace folder can be added to the application project. But whether you can place such a file under source control depends upon the source control system you are using. Some source control systems could have a problem placing a file from outside the local folder into the repository. We therefore recommend that all project files you wish to place under source control be located in the local workspace folder.

21.4 Application Project

Create or load the Altova application project you wish to place under source control. If you wish to place a single file under source control, this file must be included in a project—since source control can only be accessed via a project.

For example, consider a project in Altova's XMLSpy application. The project's properties are saved in a `.spp` file. In the application, the project is displayed in the application's Project window (*see screenshot below*). The project in the screenshot below is named `MyProject` and the project's properties are saved in the file `MyProject.spp`.



You can place the entire project (all files in the project) or only some project files under source control. **Only files that are in the project can be placed under source control.** So you will need to add files to the project before you can place them under source control. The project file (`.spp` file) will automatically be placed under source control as soon as a file from within the project is placed under source control.

The entire project, or one or more project files, is placed under source control via the command **Project | Source Control | Add to Source Control** (*see next section below*).

Note, however, that the folder structure of the repository corresponds not to the project's folder structure (*screenshot above*) but to the structure of the local workspace folder 709 (*see folder diagram below*). In the diagram below, notice that the `MyProject` folder in the repository has a folder structure corresponding to that of the local workspace folder. Note that the bound folder occurs within the repository folder.

```
Local Workspace Folder
|
|-- MyProject.spp
|-- QuickStart
|   |-- QuickStart.css
|   |-- QuickStart.xml
|   |-- QuickStart.xsd
|-- Grouping
|   |-- Persons
|   |   |-- Persons.xml
```

```
Repository
|
|-- MyProject (bound to Local Workspace)
||-- MyProject.spp
||-- QuickStart
||   |-- QuickStart.css
||   |-- QuickStart.xml
||   |-- QuickStart.xsd
||-- Grouping
||   |-- Persons
||   |   |-- Persons.xml
```

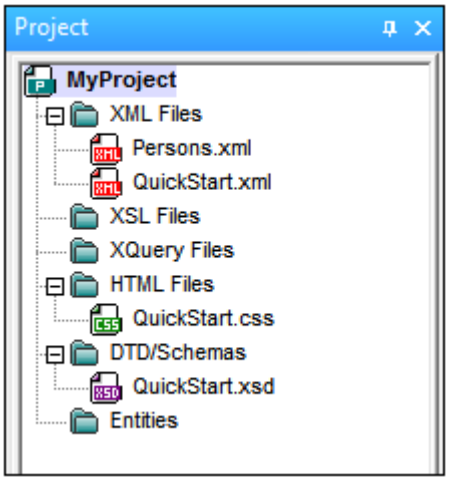
Note: An application project can contain project folders (green) and external folders (yellow). Only files in (green) project folders can be placed under source control. Files in (yellow) external folders cannot be placed under source control.

Note: Files from outside the local workspace folder can be added to the application project. But whether you can place such a file under source control depends upon the source control system you are using. Some source control systems could have a problem placing a file from outside the local folder into the repository. We therefore recommend that all project files you wish to place under source control be located in the local workspace folder.

21.5 Add to Source Control

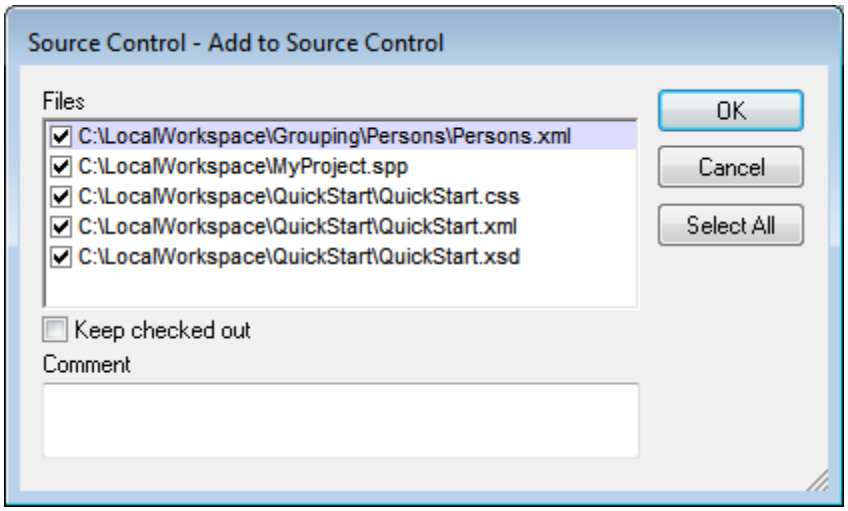
Adding the project to source control will automatically create the correct bindings and repository structure before adding the project file (`.spp` file) or individual files to source control. Add the project to source control as follows.

Select the project in the Project window (`MyProject` in the screenshot below) so that it is highlighted (*as in the screenshot below*). Alternatively select a single file, or select multiple files by clicking them with the **Ctrl** key pressed. Adding a single file to source control will automatically add the project file (`.spp` file) to source control as well.



Next, select the menu command **Project | Source Control | Add to Source Control**. This pops up the connection and configuration dialogs of the currently selected source control system. (You can change the source control system via the Change Source Control dialog (**Project | Source Control | Change Source Control**).)

Follow the source control system's instructions to make the connection and configuration. After this has been completed, all the files selected for addition plus the project file (`.spp` file) are displayed in an Add to Source Control dialog (*screenshot below*). Select the files you wish to add and click **OK**.



The files will be added to the repository and be either checked in or checked out[715] depending on whether the *Keep Checked Out* check box has been checked or not.

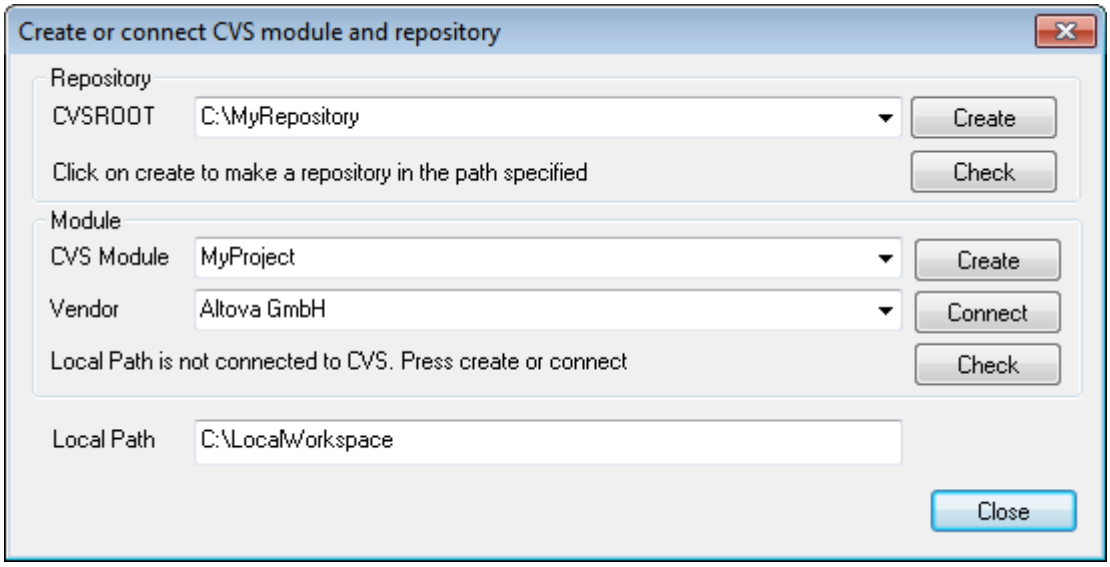
Configuration notes

You might be prompted to create a folder in the repository for the project if it has not already been created. If you are, go ahead and create it. The local workspace folder[709] will be bound to this folder created in the repository (*see diagrams below*).

```
Local Workspace Folder
|
|-- MyProject.spp
|-- QuickStart
|   |-- QuickStart.css
|   |-- QuickStart.xml
|   |-- QuickStart.xsd
|-- Grouping
|   |-- Persons
|   |   |-- Persons.xml
```

```
Repository
|
|-- MyProject (bound to Local Workspace)
||-- MyProject.spp
||-- QuickStart
||   |-- QuickStart.css
||   |-- QuickStart.xml
||   |-- QuickStart.xsd
||-- Grouping
||   |-- Persons
||   |   |-- Persons.xml
```

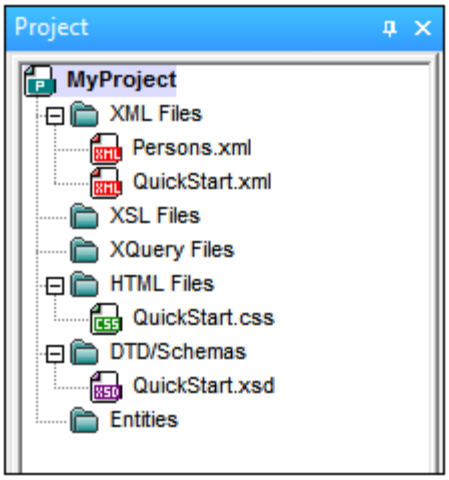
The configuration dialog of Jalindi Igloo is show below. The CVSROOT field is the path to the repository folder.



In the screenshot above, the local path locates the local workspace folder, which corresponds to the CVS module, `MyProject`, and is bound to it.

21.6 Working with Source Control

To work with source control, select the project, a project folder, or a project file in the Project window (*screenshot below*) and then select the command you want in the **Project | Source Control** menu. The **Check In** and **Check Out** commands are available as context menu commands of Project window items.



In this section, we describe the main source control features in detail:

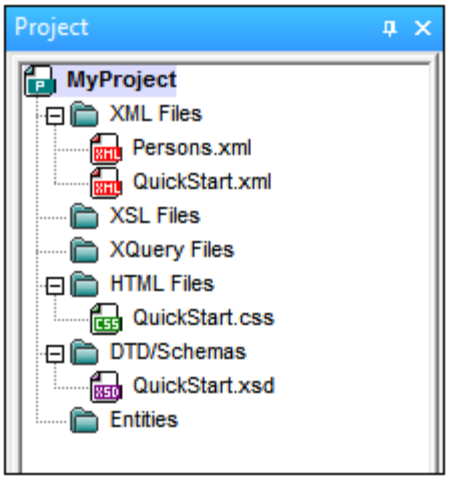
- Add to, Remove from Source Control 714
- Check Out, Check In 715
- Getting Files as Read-Only 717
- Copying and Sharing from Source Control 719
- Changing Source Control 722

Additional commands in the **Project | Source Control** menu are described in the User Reference section 785 of the manual. For information specific to a particular source control system, please see the user documentation of that system.

21.6.1 Add to, Remove from Source Control

Adding

After a project has been added to source control, you can place files either singly or in groups under source control. This is also known as adding the files to source control. Select the file in the Project window and then click the command **Project | Source Control | Add to Source Control**. To select multiple files, keep the **Ctrl** key pressed while clicking on the files you wish to add. Running the command on a (green) project folder (*see screenshot below*) adds all files in the folder and its sub-folders to source control.



When files are added to source control, the local folder hierarchy is replicated in the repository [710] (it is not the project folder hierarchy that is replicated). So, if a file is in a sub-folder X levels deep in the local folder, then the file's parent folder and all other ancestor folders are automatically created in the repository.

When the first file from a project is added to source control, the correct bindings are created in the repository and the project file (`.spp` file) is added automatically. For more details, see the section Add to Source Control [712].

Source control symbols

Files and the project folder display certain symbols, the meanings of which are given below.

	Checked in. Available for check-out.
	Checked out by another user. Not available for check-out.
	Checked out locally. Can be edited and checked-in.

Removing

To remove a file from source control, select the file and click the command **Project | Source Control | Remove from Source Control**. You can also remove: (i) files in a project folder by executing the command on the folder, and (ii) the entire project by executing the command on the project.

21.6.2 Check Out, Check In

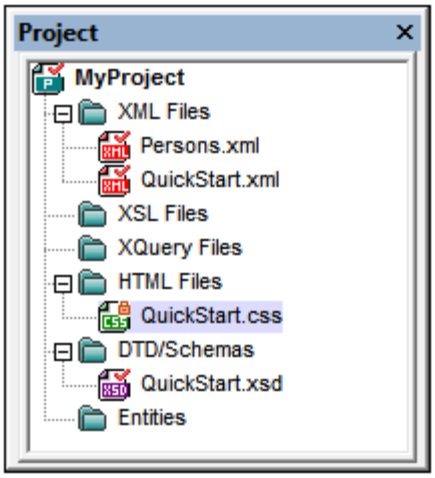
After a project file has been placed under source control, it can be checked out or checked in by selecting the file (in the Project window) and clicking the respective command in the **Project | Source Control** menu: **Check Out** and **Check In**.

When a file is checked out, a copy from the repository is placed in the local folder. A file that is checked out can be edited. If a file that is under source control is not checked out, it cannot be edited. After a file has been edited, the changes can be saved to the repository by checking in the file. Even if the file is not saved in the application, checking it in will save the changes to the repository. Whether a file is checked out or not is indicated with a tick or lock symbol in its Project window icon.

Files and the project folder display certain symbols, the meanings of which are given below.

	Checked in. Available for check-out.
	Checked out by another user. Not available for check-out.
	Checked out locally. Can be edited and checked-in.

Selecting the project or a folder within the project selects all files in the selected object. To select multiple objects (files and folders), press the **Ctrl** key while clicking the objects. The screenshot below shows a project that has been checked out. The file `QuickStart.css` has subsequently been checked in.



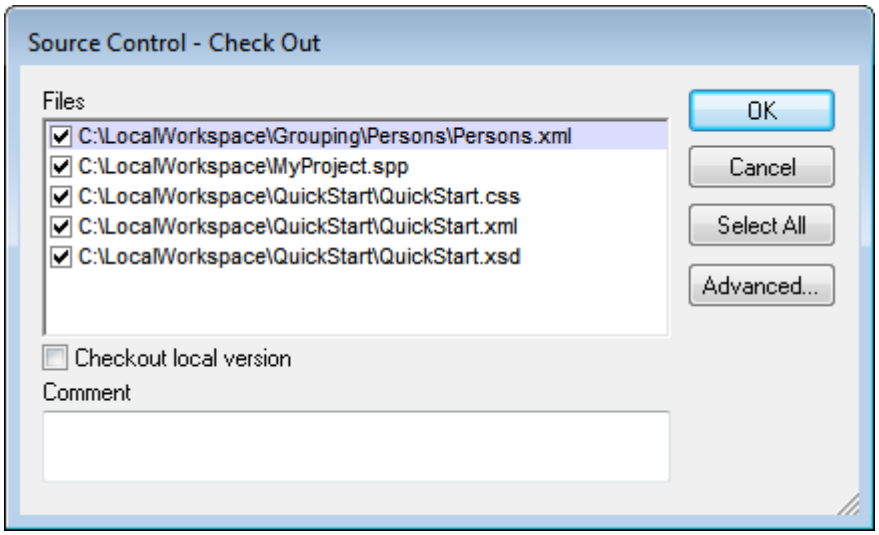
Saving and rejecting editing changes

Note that, when checking in a file, you can choose to leave the file checked out. What this does is save editing changes to the repository while continuing to keep the file checked out, which is useful if you wish to periodically save editing changes to the repository and then continue editing.

If you have checked out a file and made editing changes, and then wish to reject these changes, you can revert to the document version saved in the repository by selecting the command **Project | Source Control | Undo Check Out**.

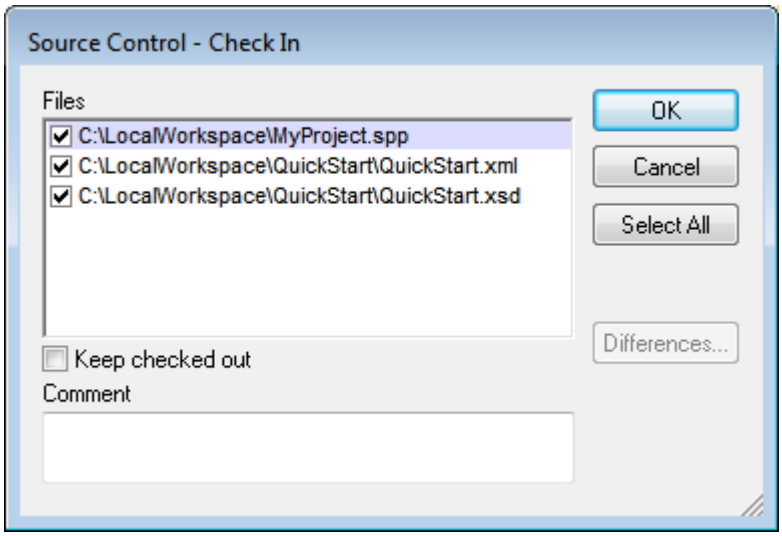
Checking out

The Check Out dialog (*screenshot below*) allows you: (i) to select the files to check out, and (ii) to select whether the repository version or the local version should be checked out.



Checking in

The Check In dialog (*screenshot below*) allows you: (i) to select the files to check in, and (ii) if you wish, to keep the file checked out.

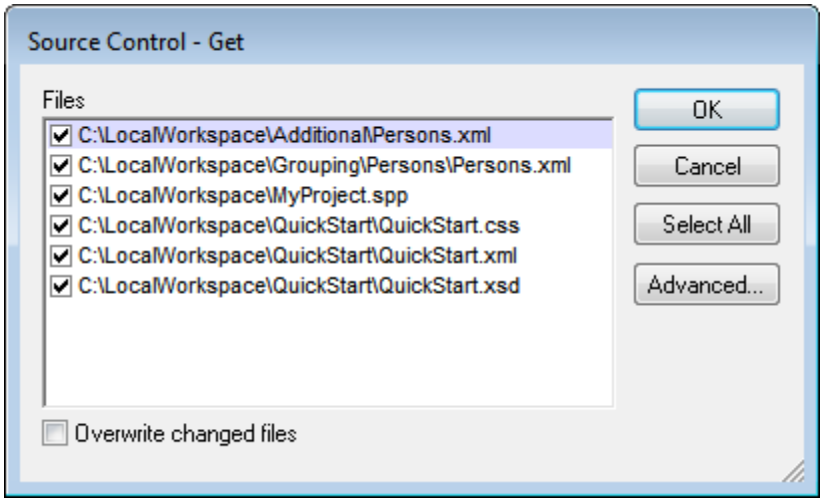


Note: In both dialogs (Check Out and Check In), multiple files appear if the selected object (project or project folder/s) contain multiple files.

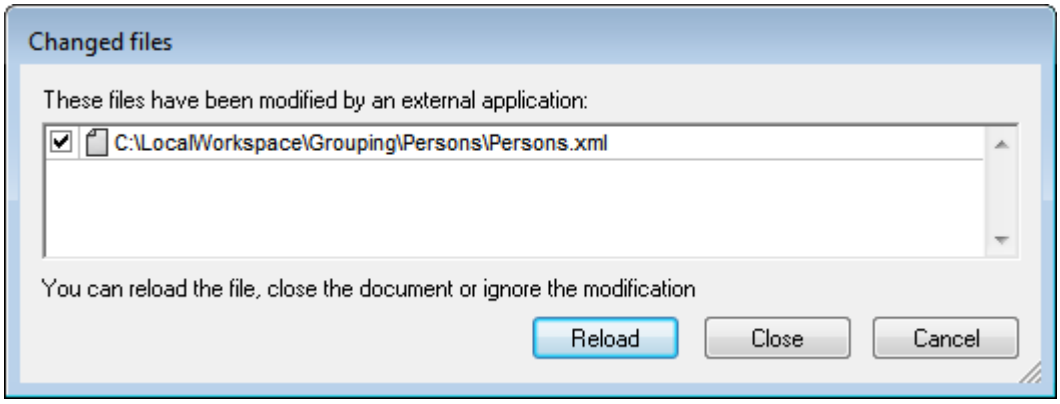
21.6.3 Getting Files as Read-Only

The **Get** command (in the **Project | Source Control** menu) retrieves files from the repository as read-only files. (To be able to edit a file, you must check it out 715.) The Get dialog lists the files in the object (project or folder) on which the **Get** command was executed (*see screenshot below*). You can select the files to retrieve by checking them in the Get dialog list.

Note: The **Get Folders** command allows you to select individual sub-folders in the repository if this is allowed by your source control system, .

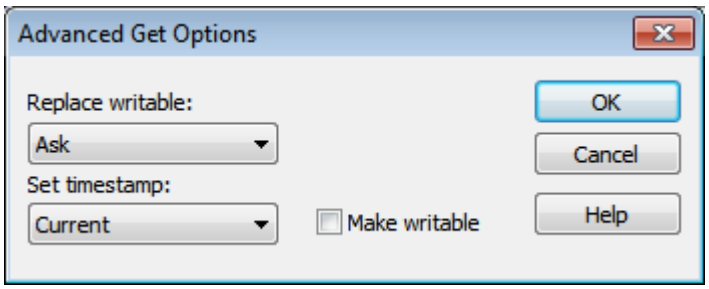


You can choose to overwrite changed checked-out files by checking this option at the bottom of the Get dialog. On clicking **OK**, the files will be overwritten. If any of the overwritten files is currently open, a dialog pops up (*screenshot below*) asking whether you wish to reload the file/s (**Reload** button), close the file/s (**Close**), or retain the current view of the file (**Cancel**).



Advanced Get Options

The Advanced Get Options dialog (*screenshot below*) is accessed via the **Advanced** button in the Get dialog (*see first screenshot in this section*).



Here you can set options for (i) replacing writable files that are checked out, (ii) the timestamp, and (iii) whether the read-only property of the retrieved file should be changed so that it will be writable.

Get latest version

The **Get Latest Version** command (in the **Project | Source Control** menu) retrieves and places the latest source control version of the selected file(s) in the working directory. The files are retrieved as read-only and are not checked out. This command works like the **Get** command (*see above*), but does not display the Get dialog.

If the selected files are currently checked out, then the action taken will depend on how your source control system handles such a situation. Typically, the source control system will ask whether you wish to replace, merge with, or leave the checked-out file as it is.

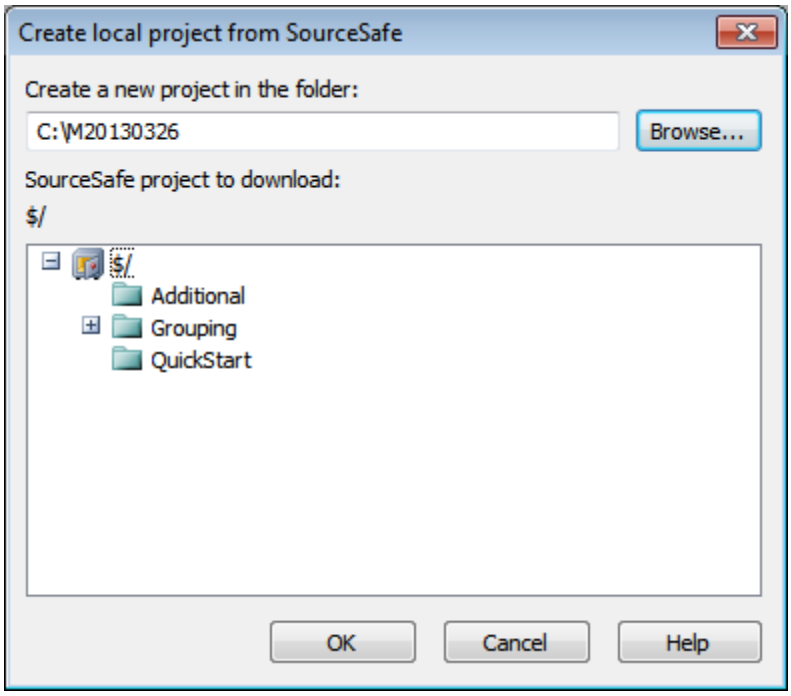
Note: This command is recursive when performed on a folder, that is, it affects all files below the current one in the folder hierarchy.

21.6.4 Copying and Sharing from Source Control

The **Open from Source Control** command creates a new application project from a project under source control.

Create the new project as follows:

1. Depending on the source control system used, it might be necessary, before you create a new project from source control, to make sure that no file from the source-controlled project is checked out.
2. No project need be open in the application, but can be.
3. Select the command **Project | Source Control | Open from Source Control**.
4. The source control system that is currently set will pop up its verification and connection dialogs. Make the connection to the bound folder in the repository 710 that you want to copy.
5. In the dialog that pops up (*screenshot below*), browse for the local folder to which the contents of the bound folder in the repository (that you have just connected to) must be copied. In the screenshot below the bound folder is called `MyProject` and is represented by the `$` sign; the local folder is `C:\M20130326`.

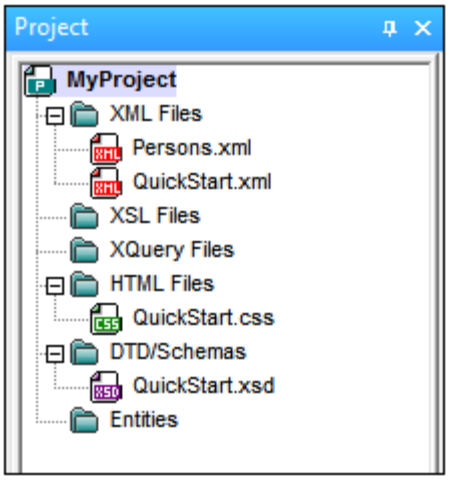


6. Click **OK**. The contents of the bound folder (`MyProject`) will be copied to the local folder `C:\M20130326`., and a dialog pops up asking you to select the project file (`.spp` file) that is to be created as the new project.
7. Select the `.spp` file that will have been copied to the local folder. In our example, this will be `MyProject.spp` located in the `C:\M20130326` folder. A new project named `MyProject` will be created in the application and will be displayed in the Project window. The project's files will be in the folder `C:\M20130326`.

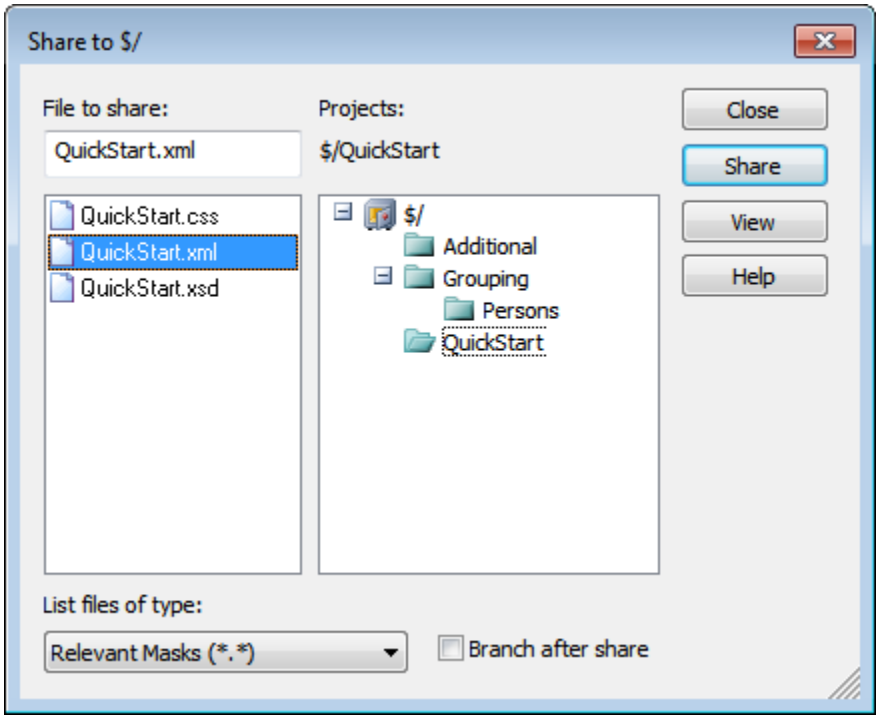
Sharing from source control

The **Share from Source Control** command is supported when the source control system being used supports shares. You can share a file, so that it is available at multiple local locations. A change made to one of these local files will be reflected in all the other "shared" versions.

In the application's Project window first select the project (*highlighted in the screenshot below*). Then click the **Share from Source Control**.

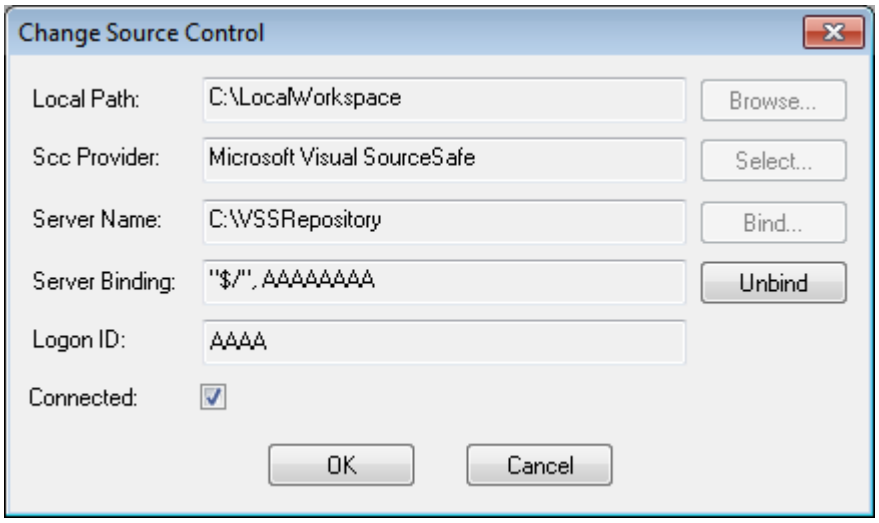


The Share To [Folder] dialog (*screenshot below*) pops up.



To select the files to share, first choose, in the project tree in the right-hand pane of the dialog (*see screenshot above*), the folder in which the files are. The files in the chosen folder are displayed in the left-hand pane. Select the file you wish to share (multiple files by pressing the **Ctrl** key and clicking the files you want to share). The selected file/s will be displayed in the *Files to Share* text box (*at top left*). The files disappear from the left hand pane. Click **Share** and then **Close** to copy the selected file/s to the local share folder. When you click **Close**, the files to share will be copied to the selected local location.

The share folder is noted in the name of the Share to [Folder] dialog. In the screenshot above it is the local folder (since the $ sign is the folder in the repository to which the local folder is bound). You can see and set the share folder in the Change Source Control dialog (*screenshot below*, **Change Source Control**) by changing the local path and server binding.



For more details about sharing using your source control system, see the source control system's user documentation.

21.6.5 Changing Source Control

Source control settings can be changed via two commands in the **Project | Source Control** menu:

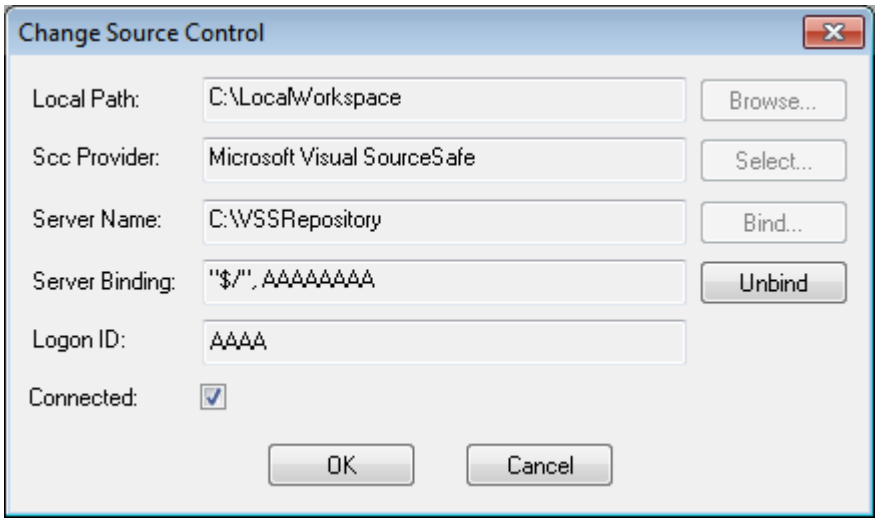
- **Source Control Manager**, which opens the source control system application and allows you to set up databases and configure bindings.
- **Change Source Control**, which pops up the Change Source Control dialog, in which you can change the source control system being used by the Altova application and the current binding. This dialog is described below.

The current binding is what the active application project will use to connect to the source control database. The current binding is correct when the application project file (`.spp` file) is in the local folder and the bound folder in the repository is where this project's files are stored. Typically the bound folder and its sub-structure will correspond with the local workspace folder and its sub-structure.

In the Change Source Control dialog (*screenshot below*), you can change the source control system (*SCC Provider*), the local folder (*Local Path*), and the repository binding (*Server Name* and *Server Binding*).

Only after undoing the current binding can the settings be changed. Undo the current binding with the **Unbind** button. All the settings are now editable.

 (C) 2015-2021 Altova GmbH



Change source control settings as follows:

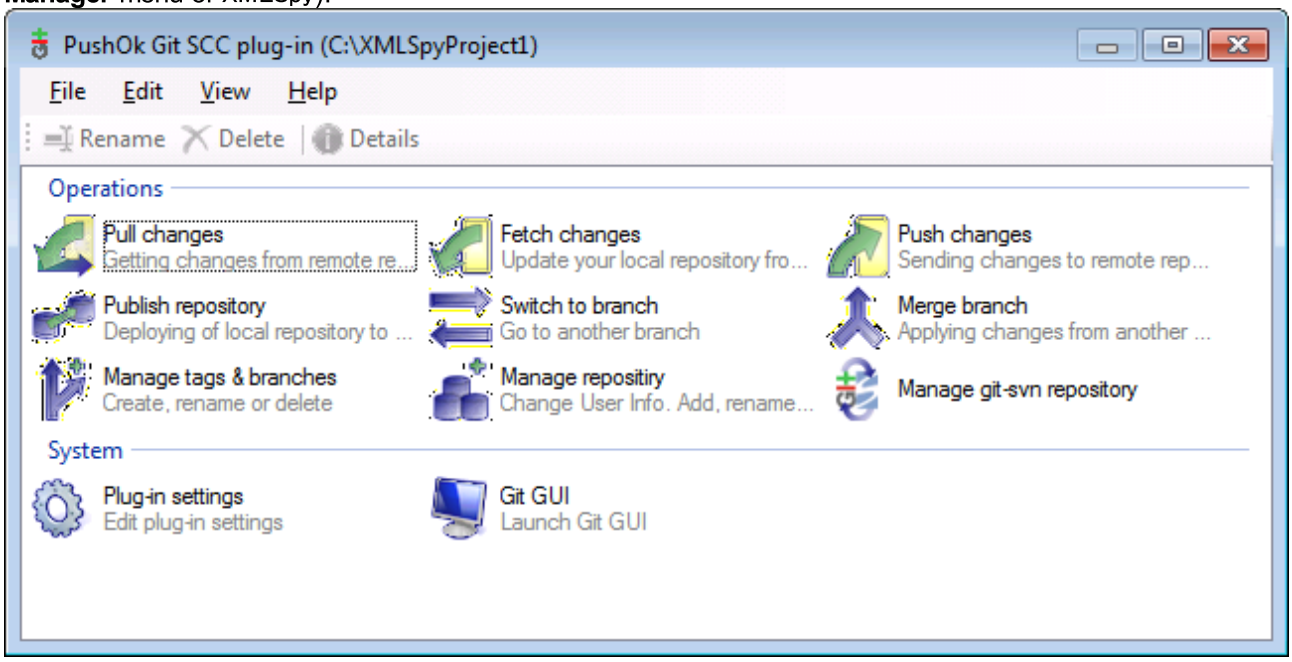
1. Use the **Browse** button to browse for the local folder and the **Select** button to select from among the installed source control systems.
2. After doing this you can bind the local folder to a repository database. Click the **Bind** button to do this. This pops up the connection dialog of your source control system.
3. If you have entered a *Logon ID*, this will be passed to the source control system; otherwise you might have to enter your logon details in the connection dialog.
4. Select the database in the repository that you wish to bind to this local folder. This setting might be spread over more than one dialog.
5. After the setting has been created, click **OK** in the Change Source Control dialog.

21.7 Source Control with Git

Support for Git as a source control system in XMLSpy is available through a third-party plug-in called **GIT SCC plug-in** (http://www.pushok.com/software/git.html).

> At the time when this documentation is written, the **GIT SCC plug-in** is available for experimental use. Registration with the plug-in publisher is required in order to use the plug-in.

The GIT SCC plug-in enables you to work with a Git repository using the commands available in the **Project | Source Control** menu of XMLSpy. Note that the commands in the **Project | Source Control** menu of XMLSpy are provided by the Microsoft Source Control Plug-in API (MSSCCI API), which uses a design philosophy different from Git. As a result, the plug-in essentially mediates between "Visual Source Safe"-like functionality and Git functionality. On one hand, this means that a command such as **Get latest version** may not be applicable with Git. On the other hand, there are new Git-specific actions, which are available in the "Source Control Manager" dialog box provided by the plug-in (under the **Project | Source Control | Source Control Manager** menu of XMLSpy).



The Source Control Manager dialog box

Other commands that you will likely need to use frequently are available directly under the **Project | Source Control** menu.

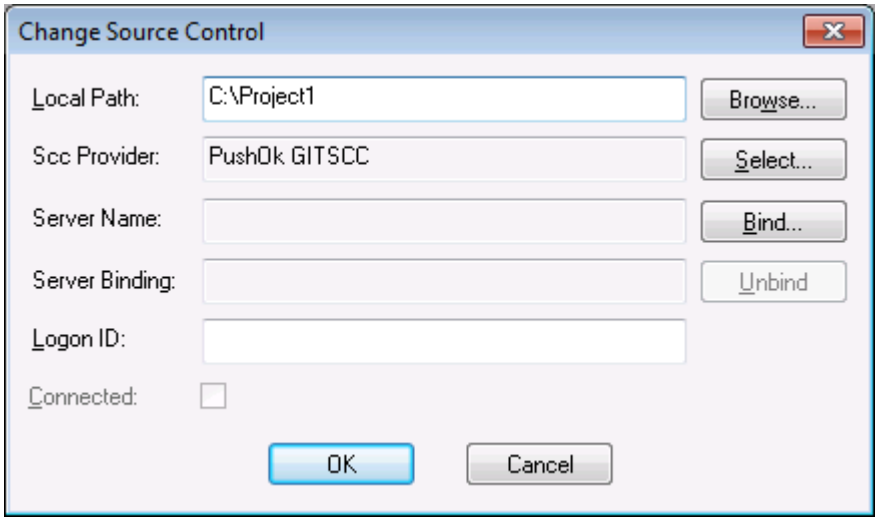
The following sections describe the initial configuration of the plug-in, as well as the basic workflow:

- Enabling Git Source Control with GIT SCC Plug-in 725
- Adding a Project to Git Source Control 725
- Cloning a Project from Git Source Control 726

21.7.1 Enabling Git Source Control with GIT SCC Plug-in

To enable Git source control with XMLSpy, the third-party **PushOK GIT SCC plug-in** must be installed, registered, and selected as source control provider, as follows:

1. Download the plug-in installation file from the publisher's website (http://www.pushok.com), run it, and follow the installation steps.
2. On the **Project** menu of XMLSpy, click **Change Source Control**, and make sure **PushOk GITSCC** is selected as source control provider. If you do not see **Push Ok GITSCC** in the list of providers, it is likely that the installation of the plug-in was not successful. In this case, check the publisher's documentation for a solution.



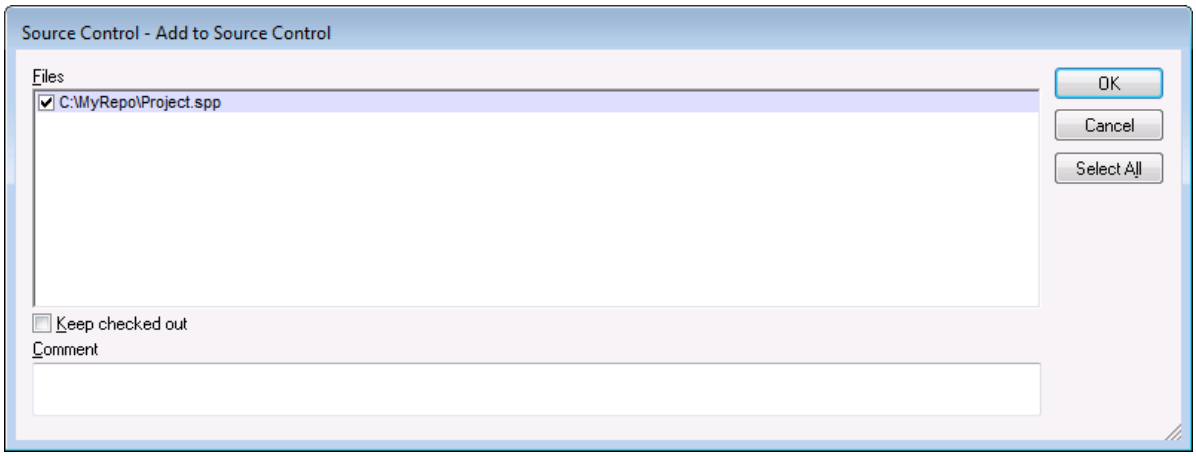
3. When a dialog box prompts you to register the plug-in, click **Registration** and follow the wizard steps to complete the registration process.

21.7.2 Adding a Project to Git Source Control

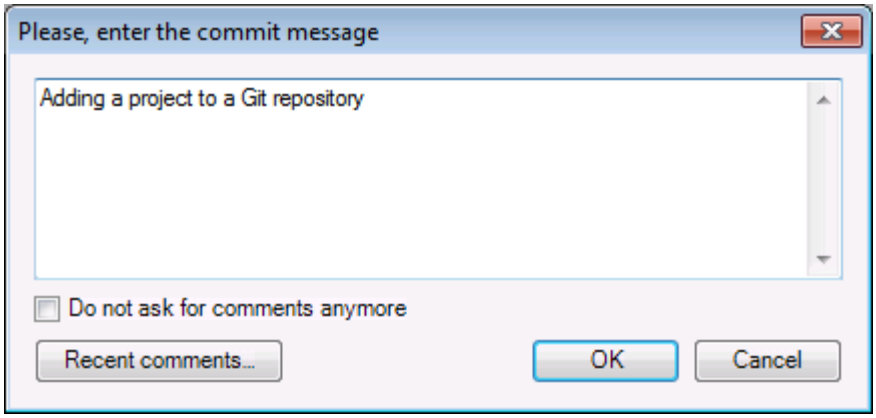
You can save XMLSpy projects as Git repositories. The structure of files or folders that you add to the project would then correspond to the structure of the Git repository.

To add a project to Git source control:

1. Make sure that **PushOK GIT SCC Plug-in** is set as source control provider (see Enabling Git Source Control with GIT SCC Plug-in 725).
2. Create a new project using the menu command **Project | Create Project**.
3. Save the project to a local folder, for example `C:\MyRepo\Project.spp`
4. On the **Project** menu, under **Source Control**, click **Add to Source Control**.

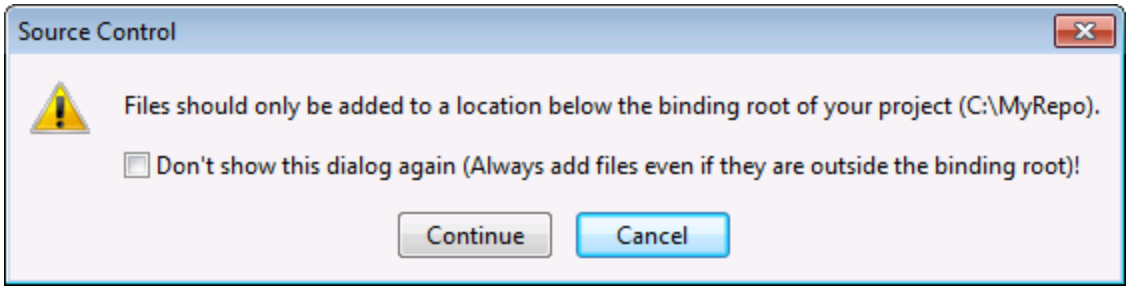


5. Click **OK**.



6. Enter the text of your commit message, and click **OK**.

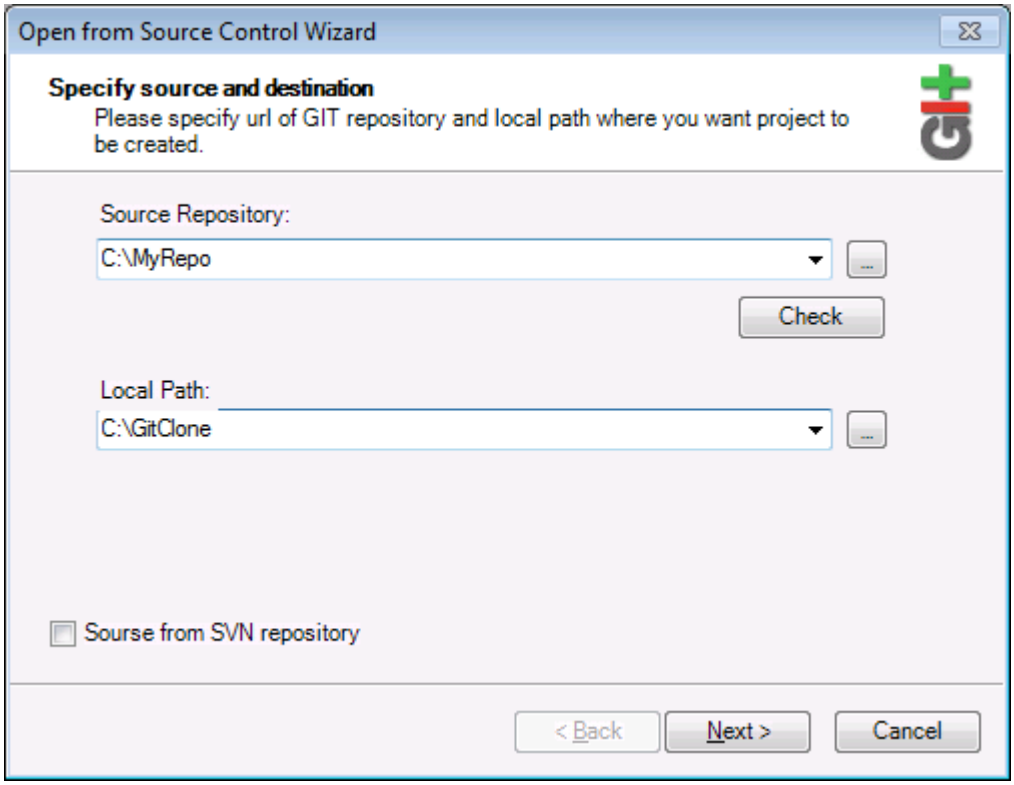
You can now start adding files and folders to your project. Note that all project files and folders must be under the root folder of the project. For example, if the project was created in the `C:\MyRepo` folder , then only files under `C:\MyRepo` should be added to the project. Otherwise, if you attempt to add to your project files that are outside the project root folder, a warning message is displayed:



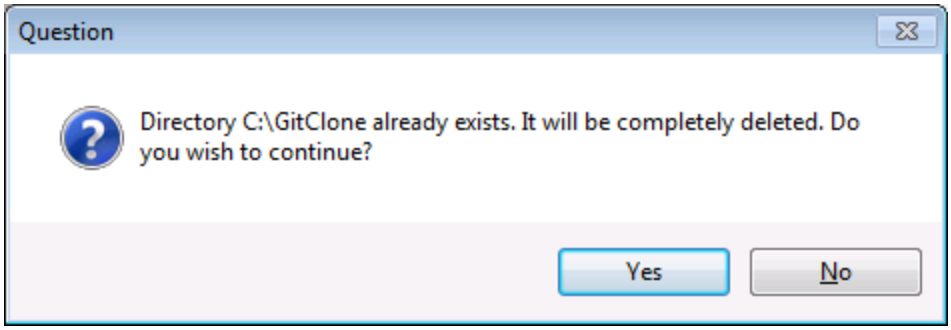
21.7.3 Cloning a Project from Git Source Control

Projects that have been previously added to Git source control (see Adding a Project to Git Source Control 725) can be opened from the Git repository as follows:

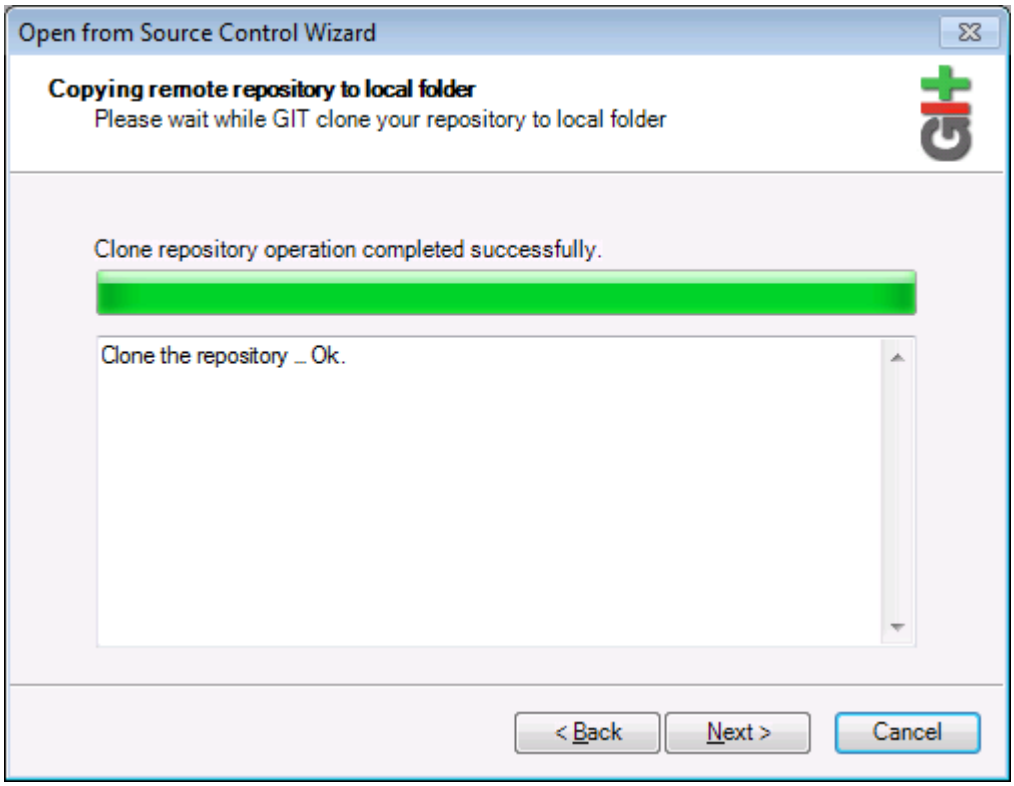
1. Make sure that PushOK GIT SCC Plug-in is set as source control provider (see Enabling Git Source Control with GIT SCC Plug-in 725).
2. On the **Project** menu, click **Source Control | Open from Source Control**.
3. Enter the path or the URL of the source repository. Click **Check** to verify the validity of the path or URL.



4. Under **Local Path**, enter the path to local folder where you want the project to be created, and click **Next**. If the local folder exists (even if it is empty), the following dialog box opens:



5. Click **Yes** to confirm, and then click **Next**.



6. Follow the remaining wizard steps, as required by your specific case.
7. When the wizard completes, a Browse dialog box appears, asking you to open the XMLSpy Project (*.spp) file. Select the project file to load the project contents into XMLSpy.

22 Visual Studio 内の XMLSpy

XMLSpy はMicrosoft Visual Studio IDE のバージョン 2010/2012/2013/2015/2017/2019 に統合することができます。この機能により、高度な XML 編集機能を、Visual Studio の高度な開発環境内で使用することができるようになります。

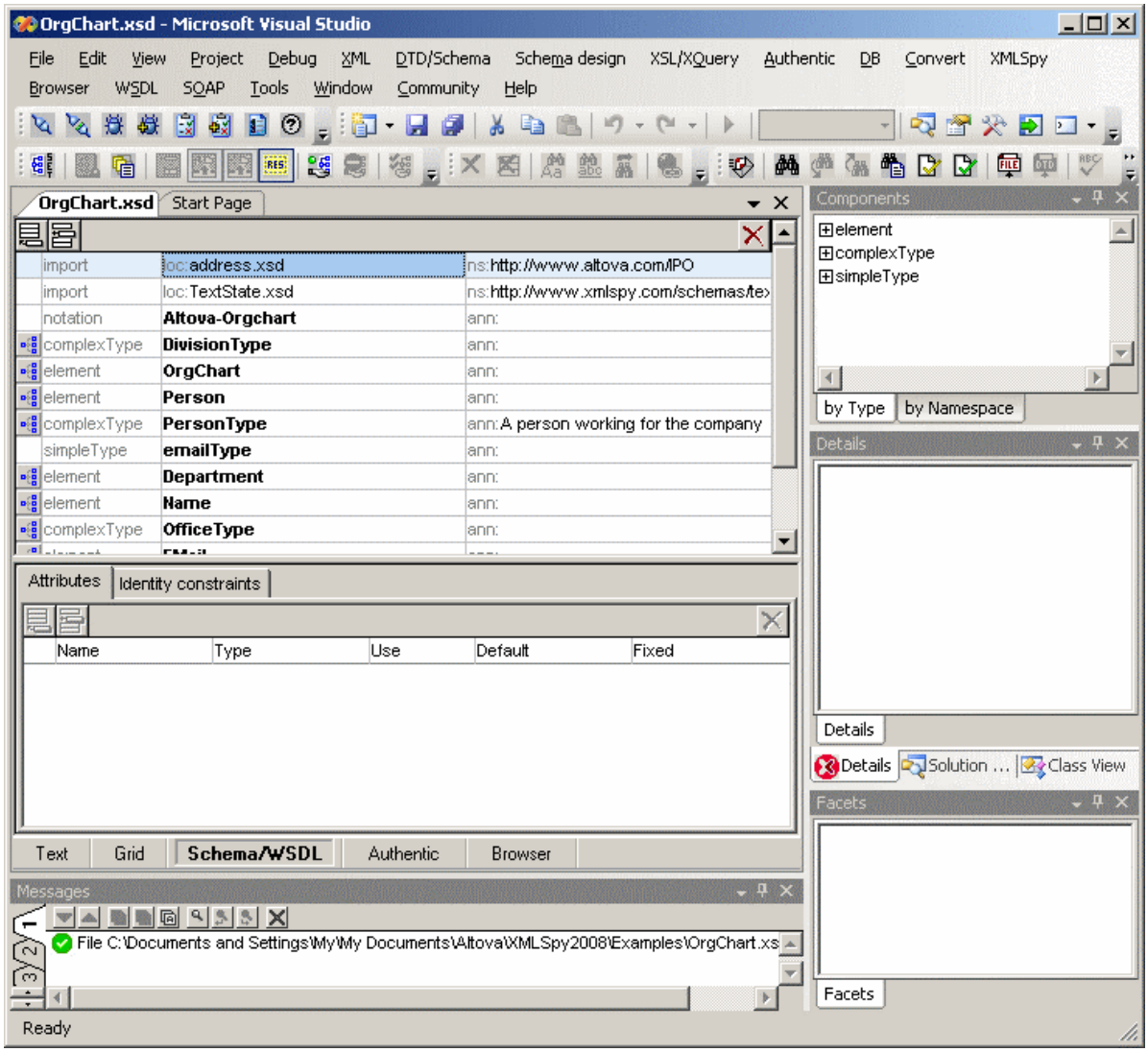
このセクションでは、以下のように構成されます:

- 大まかなインストールの流れ[730]と、XMLSpy プラグインをVisual Studio へ統合
- Visual Studio バージョンとスタンドアロンバージョンの違い[731]
- Visual Studio 内のXMLSpy デバッガー[733]

22.1 XMLSpy プラグインのインストール

Visual Studio で使えるXMLSpy プラグインをインストールするには、以下の手順を踏みます:

1. Microsoft Visual Studio 2010/2012/2013/2015/2017/2019 のインストール。
2. XMLSpy (Enterprise またはProfessional Edition) のインストール
3. XMLSpy のMicrosoft Visual Studio Integration Package をダウンロードして実行。パッケージはwww.altova.com にあるXMLSpy (Enterprise またはProfessional Edition)ダウンロードページにてご利用になれます。(注意: XMLSpy バージョンに対応するIntegration Package を使用する必要があります(現在のバージョンは2021 です)。)



Integration Package をインストールすると、Visual Studio 環境からXMLSpy を使用することができるようになります。

22.2 XMLSpy スタンドアロンとの違い

このセクションではVisual Studio バージョンとスタンドアロンバージョンのXMLSpy の違いについて説明します。まずはVisual Studio バージョンでサポートされていない機能について、そしてスタンドアロンバージョンのVisual Studio 間で動作が異なる点について説明します。

- Visual Studio ではサポートされない機能 731
- Visual Studio で追加されたXMLSpy メニュー 731
- 入力ヘルパー(Visual Studio のツールウィンドウ) 731
- 異なるコマンドによる同等の機能 731
- Visual Studio コマンドとして使えるXMLSpy コマンド 732

Visual Studio ではサポートされない機能

Visual Studio では以下のXMLSpy の機能がご利用になれません:

- スクリプティング環境(「**ツール | XMLSpy オプション**」メニューオプションの「**スクリプト**」タブ)は現在サポートされていません。
- メニューオプションの「**ツール | ツール**」の「**ビュー**」タブにある、ブラウザービューを別のウィンドウに表示するオプションがサポートされていません。つまりテキストビューとテキストビューは常に同じウィンドウで表示されます。
- Authentic View 401 の状態アイコンがサポートされていません。
- 全てのソース管理機能がサポートされていません。
- (スタンドアロンバージョンでは「**ツール**」メニューオプションから利用できる)全ての比較機能がサポートされていません。

Visual Studio で追加されたXMLSpy メニュー

以下のコマンドはVisual Studio バージョンのXMLSpy だけでご利用になれます:

- **表示 | XMLSpy ツールウィンドウ**
- **表示 | XMLSpy 表示**
- **XMLSpy** (グローバルリソースメニューアイテムが含まれます)
- ツール| XMLSPY オプション

入力ヘルパー(Visual Studio のツールウィンドウ)

XMLSpy の入力ヘルパーはVisual Studio のツールウィンドウとして使用することができます。入力ヘルパーに関して、以下の点に注意してください。入力ヘルパーとXMLSpy のGUI に関しては、イントロダクション 107 を参照ください。

- 入力ヘルパーウィンドウは開発環境内のどのような場所にでもドラッグすることができます。
- 入力ヘルパータブを右クリックすることで、インターフェースを更にカスタマイズすることができます。入力ヘルパーのオプションは: フローティング、ドッキング可能、タブ付きドキュメント、自動的に隠す、です。

異なるコマンドによる同等の機能

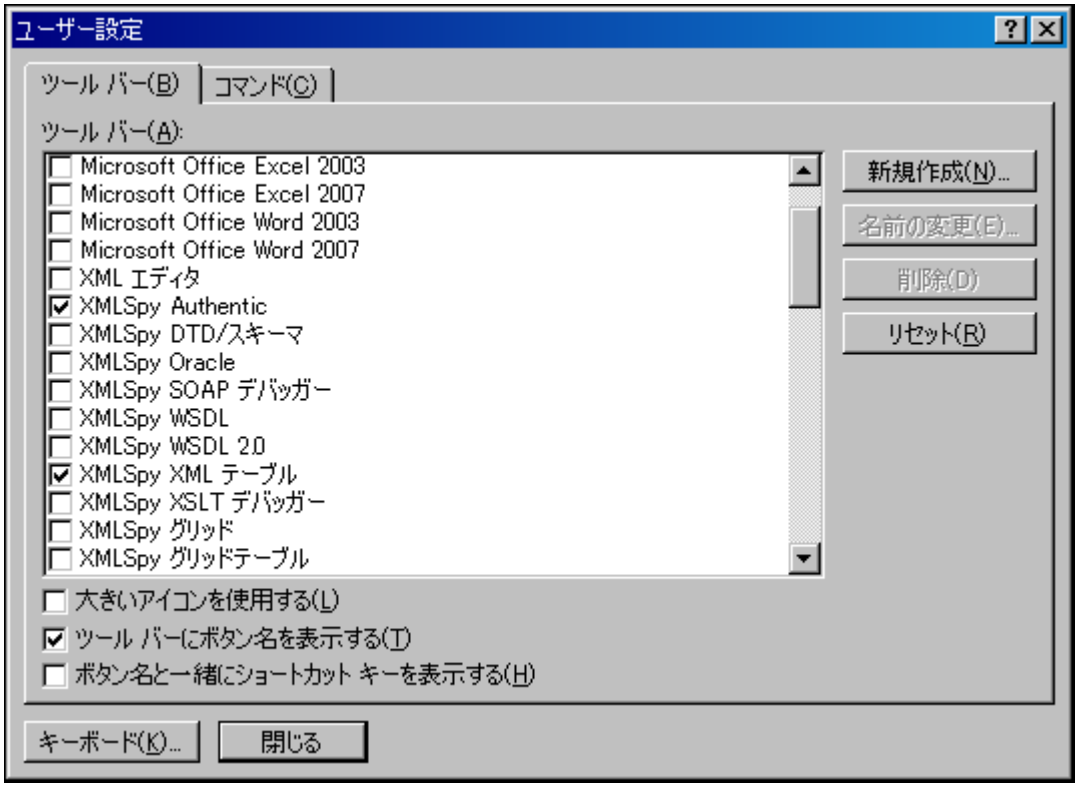
以下の表に示されるように、XMLSpy の機能によっては、Visual Studio にて異なるコマンド名で表示されます。

XMLSpy	*Visual Studio*	*機能*
ファイル| 開く	URL にスイッチ	ファイル| 開く	W eb サイト	URL からファイルを開く
ファイル| 名前を付けて保存	URL にスイッチ	ファイル| XMLSpy ファイルを URL に保存	ファイルを URL に保存

Visual Studio コマンドとして使える XMLSpy コマンド

いくつかのXMLSpy コマンドはVisual Studio コマンドとしてVisual Studio GUI に表示されます:

- *元に戻す、やり直し* :これらのVisual Studio コマンドは、Visual Studio 開発環境全てに対して適用されます。
- *プロジェクト:* XMLSpy プロジェクトはVisual Studio プロジェクトとして扱われます。
- *ツールバーのカスタマイズ、コマンドのカスタマイズ* :(以下のスクリーンショットにある)ユーザー設定ダイアログ(「**ツール | ユーザー設定**」)内のツールバーならびにコマンドタブにはVisual Studio とXMLSpy のコマンドが含まれます。

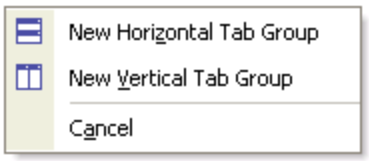


- *表示* :**表示**メニュー以下のXMLSpy ツールウィンドウとXMLSpy ビューの2つのコマンドには、入力ヘルパーや他のサイドバー、編集ビュー間の切り替え、そして特定の入力ガイドの有効/無効を切り替えるオプションが含まれています。
- *XMLSpy ヘルプ:* このXMLSpy メニューはVisual Studio「**ヘルプ**」メニューのサブメニューに表示されます。

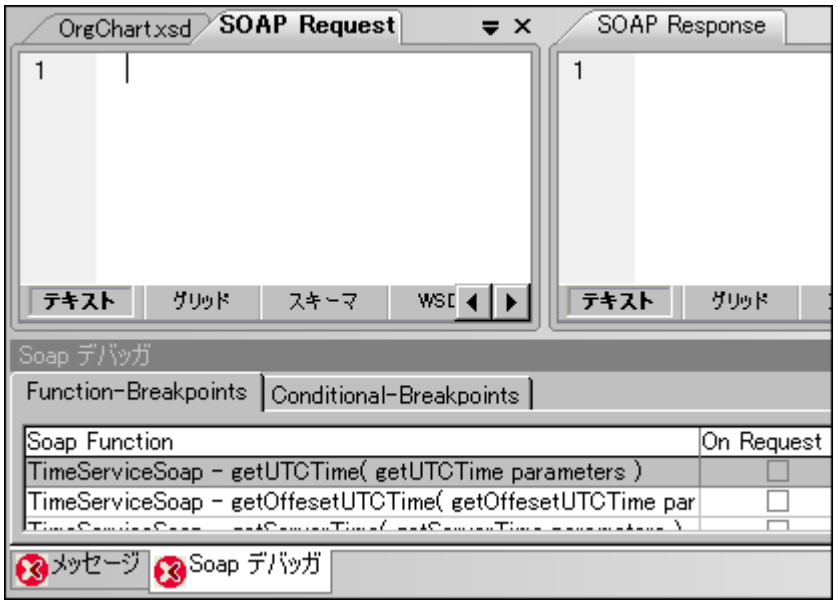
22.3 Visual Studio での XMLSpy デバッガー

XMLSpy にはXSLT/XQuery デバッガー(Enterprise ならびにProfessional Edition)と、SOAP デバッガー(Enterprise Edition)が搭載されています。デバッガーでは(例えばXML、XSLT、XSLT 出力ファイルなど)複数のファイルが表示されますが、これらのファイルはVisual Studio にて1つにタブグループ化されます。Visual Studio では新たにタブグループを作成して、デバッグを見やすく、処理の経過をより簡単にたどることができます:

1. 単一のタブグループから分離したいタブをクリックします。そのタブを、現在アクティブなタブのどこかにドラッグします。この操作により、新たに作成するタブグループの種類を決定するポップアップメニューが表示されます。



2. 「**垂直タブグループの新規作成**」を選択すると、選択されたタブのみが含まれるタブグループが新たに作成されます(以下のスクリーンショットを参照)。



23 Eclipse 内の XMLSpy

Eclipse はオープンソースのフレームワークで、様々なアプリケーション開発環境をプラグインという形で統合します。

Eclipse 用のXMLSpy プラグインを使えば、Eclipse 2020-12, 2020-09, 2020-06, 2020-03 プラットフォームからXMLSpy の機能にアクセスすることができるようになります。プラグインはW indows プラットフォームにてご利用になれます。このセクションではEclipse 用のXMLSpy プラグインのインストール方法[735]と、XMLSpy パースペクティブ[737]ならびにXMLSpy デバッガーパースペクティブ[740]の設定方法について説明します。これらの操作を行うと、XMLSpy GUI とXMLSpy メニューコマンドをEclipse GUI から使用することができるようになります。

注意: スタンドアロンバージョンで使用することのできるソース管理機能はEclipse バージョンにてご利用になれません。

(C) 2015-2021 Altova GmbH

23.1 Eclipse 用の XMLSpy プラグインをインストール

必要条件

- Eclipse 2020-12, 2020-09, 2020-06, 2020-03 (http://www.eclipse.org)。
- the 64-ビットプラットフォームのためのJava Runtime Environment（JRE）またはJava Development Kit（JDK）
- XMLSpy Enterprise またはProfessional Edition 64-ビット
- XMLSpy Integration Package 64-ビット を以下でダウンロードすることができます https://www.altova.com/ja/components/download。下記の通りEclipse 統合をインストールの一部として行うことができます。

メモ　上記の必要条件は64-ビットプラットフォームを必要としています。古いEclipse 32-ビットプラットフォームとの統合は作動する場合がありますが、サポートされていません。

Eclipse のための XMLSpy プラグインをインストールする方法

Eclipse のための XMLSpy プラグインの統合を以下の方法の一つで行うことができます:

1. 自動的にXMLSpy Integration Package のインストール中に統合することができます（これは奨励されるオプションです）。
2. XMLSpy Integration Package を最初にインストールし、Eclipse からプラグインを手動で統合する。

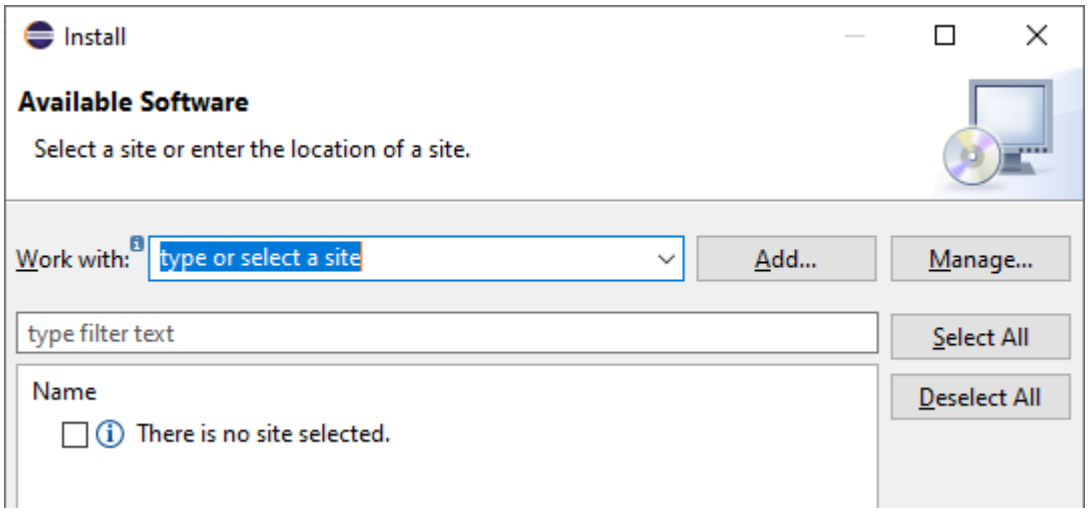
Eclipse のための XMLSpy プラグインを統合する方法:

1. インストールウィザードを開始するためにXMLSpy Integration Package を実行します。
2. プロンプトされると *Eclipse プラグインのインストール* オプションを選択し [次へ]. をクリックします。
3. プロンプトされるとXMLSpy プラグインのEclipse への統合方法を選択します。以下の1つを行ってください:
   - プラグインインストールを自動的に完了する場合 *このウィザードに Altova XMLSpy プラグインを Eclipse に統合する*を選択し、Eclipse 実行可能ファイル（`eclipse.exe`）がロードされている場所を参照します。
   - Eclipse 内で後にプラグインのインストールを完了する場合、clear the *このウィザード...* チェックボックスをクリアします。
4. [次] をクリックすることでインストールを完了します。自動統合を選択する場合 XMLSpy パースペクティブとメニューがEclipse を次回起動する際にEclipse 内で使用可能になります。

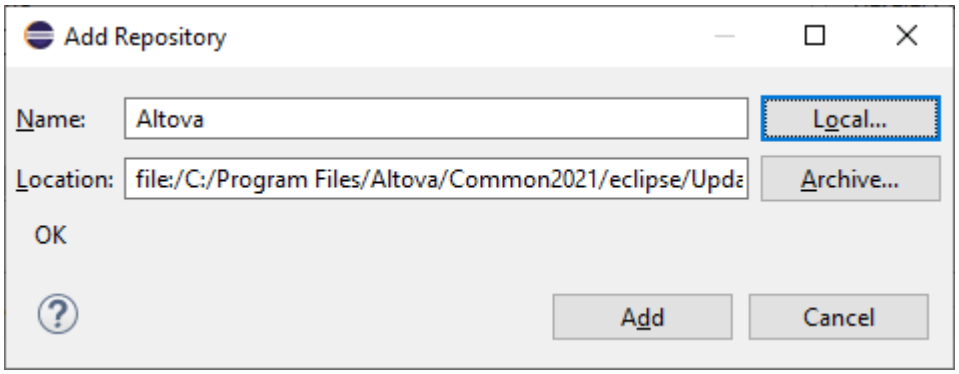
メモ　XMLSpy Integration Package をインストールまたはアンインストールする場合 Eclipse を閉じる必要があります。

Eclipse のための XMLSpy プラグインを手動で統合する方法

1. Eclipse にてメニューオプションから[**ヘルプ | 新規ソフトウェアのインストール**]を選択します。
2. [インストール] ダイアログボックス内で [**追加**]をクリックします。



3. [レポジトリの追加] ダイアログボックス内で [ローカル] をクリックします。フォルダー `C:\Program Files\Altova\Common2021\eclipse\UpdateSite` を参照して選択します。("Altova" などの) サイトの名前を与えます。



4. ステップ 2 から 3 を繰り返します、今度は、フォルダー `C:\Program Files\Altova\XMLSpy2021\eclipse\UpdateSite` を選択し、XMLSpy などの名前を与えます。
5. インストールダイアログボックスで *ローカルのサイトのみ* を選択します。次に [Altova category] フォルダーを選択し [次へ] をクリックします。
6. インストールされるアイテムをレビューし [次へ] をクリックして継続します。
7. 使用許諾契約書に合意するには対応するチェックボックスを選択します。
8. [完了] をクリックすることでインストールを完了します。

メモ (不足するアイコンなど) プラグインの使用に関する問題が発生した場合 `-clean` プラグを使用してコマンドラインから Eclipse を起動します。

(C) 2015-2021 Altova GmbH

23.2 Eclipse 内の XMLSpy エントリポイント

Eclipse 内にある以下のエントリポイントを使って XMLSpy の機能にアクセスすることができます:

- XMLSpy パースペクティブ 737 から XMLSpy の GUI 機能に Eclipse GUI からアクセスすることができます。
- XMLSpy ツールバーボタン 739 から XMLSpy のヘルプならびにドキュメントの新規作成機能にアクセスすることができます。

XMLSpy パースペクティブ

Eclipse における パースペクティブとは、特定の機能に結び付けられた GUI ビューのことです。Eclipse 用の XMLSpy プラグインが Eclipse に統合されると、XMLSpy パースペクティブも自動的に作成されます。この パースペクティブは、編集ビュー、メニュー、入力ヘルパー、そしてサイドバーなど、XMLSpy の GUI 要素を含む パースペクティブとなります。

XMLSpy と関連付けられている種類のファイル（例えば `.xml`）が開かれた時、このファイルを XMLSpy パースペクティブで編集することができます。同様に、他の種類のファイルも Eclipse の別の パースペクティブで開くことができます。更に、アクティブなファイルに対しては、パースペクティブを変更することができ、異なる環境で同じファイルの編集を行うことができます。従って、パースペクティブには2つの利点があります:

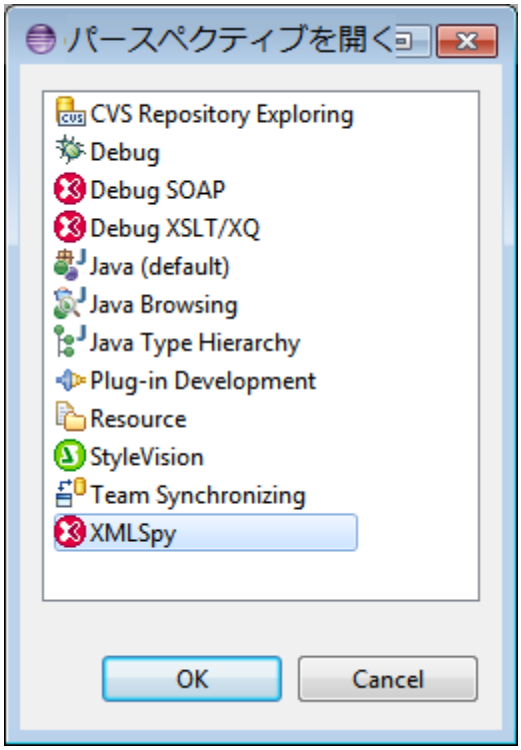
1. アクティブなファイルに対して、作業環境を素早く変更することができる。
2. 新たな開発環境を開くこと無くファイルの切り替えを行うことができる（パースペクティブに結び付けられた環境も有効になります）。

XMLSpy パースペクティブを使った作業には以下のようなものがあります:

- XMLSpy パースペクティブへの切り替え
- XMLSpy パースペクティブの環境設定
- XMLSpy パースペクティブのカスタマイズ

XMLSpy パースペクティブへの切り替え

Eclipse にて、**メニューコマンド** から「**ウィンドウ | パースペクティブ | パースペクティブを開く | その他**」を選択します。表示されるダイアログボックス内で（*下のスクリーンショット*）select **XMLSpy** を選択して［**開く**］をクリックします。



空のドキュメント、またはアクティブなドキュメントに対して XMLSpy パースペクティブが開かれます。この方法でユーザーはパースペクティブの切り替えを行うことができます。目的のパースペクティブをパースペクティブを開くサブメニュー（のその他アイテムの上）に表示させることで、パースペクティブにより素早くアクセスすることができます。

この設定はカスタマイズダイアログにて行うことができます。ファイルが開かれた時、またはアクティブになったときにパースペクティブを自動的に切り替えることもできます。そのファイルが最初に開かれたときには、ファイルの種類に関連付けられているアプリケーションのパースペクティブが自動的に開かれ、このファイルの種類に対して、このパースペクティブをデフォルトのパースペクティブとして割り当てるか尋ねられます。この種類のファイルを開くたびに尋ねられたくない場合は、*今後このメッセージを表示しない* にチェックを入れ「OK」をクリックします。

XMLSpy パースペクティブの環境設定

パースペクティブの優先順位には以下が含まれています: XMLSpy パースペクティブの環境設定パースペクティブの設定には、(i) 関連付けられた種類のファイルが開かれたときに、自動的にパースペクティブを切り替える、（*上の参照*）(ii) 各 XMLSpy ツールバーを含めるオプションが含まれます。設定ダイアログにアクセスするには、[**ウィンドウ | 設定**] メニューコマンドを選択します。左のペインにあるパースペクティブのリストから XMLSpy を選択し、必要な設定を選択します。「OK」ボタンをクリックして完了します。

XMLSpy パースペクティブのカスタマイズ

カスタマイズオプションを使うことで、ショートカットやコマンドをパースペクティブに加えることができます。（*以下のスクリーンショットにある*）パースペクティブのカスタマイズダイアログにアクセスするには、（*XMLSpy パースペクティブ*の場合は）パースペクティブをアクティブにして、メニューコマンドから[**ウィンドウ | パースペクティブ | パースペクティブのカスタマイズ**]を選択します。
ツールバー可視性とメニュー可視性タブ内では、度のツールバーとメニューが表示されるかを指定することができます。アクションセットの使用可能タブでは、親メニューとツールバーにアクションセットを追加することができます。アクショングループ可用性タブでは、アクションセットを親メニューやツールバーへ追加することができます。アクショングループを有効にするには、対応するチェックボックスにチェックを入れてください。カスタマイズパースペクティブダイアログのショートカットタブでは、サブメニューのショートカットをセットすることができます。サブメニューコンボボックスから目的のサブメニューを選択し、ショートカットのカテゴリを選択し、パースペクティブに追加したいショートカットにチェックを入れます。[**適用 & 閉じる**] をクリックしてカスタマイズを完了すると変更が反映されます。

XMLSpy ツールバーボタン

The **XMLSpy** メニューにはXMLSpy により認識されるドキュメントの型が現在 Eclipse 内で開かれていない場合でも関連性のあるコマンドが含まれています。XMLSpy のスタンダードバージョンではコマンドの一部は「ファイル」メニュー内に存在しています。

XMLSpy ツールバーには次のボタンが含まれています（*下のスクリーンショット参照*）。

これらは左側から：(i) XMLSpy ヘルプを開く (ii) (**XMLSpy** メニューからアクセスすることの代替として)XMLSpy コマンドにアクセス

23.3 XMLSpy のデバッガーパースペクティブ

Enterprise Edition のXMLSpy には2つのデバッガー(XSLT/XQuery ならびにSOAP)が搭載されており、Professional Edition のXMLSpy には1つのデバッガー(XSLT/XQuery)が搭載されています。これらデバッガーのパースペクティブも、現在インストールされているXMLSpy のエディションに応じてEclipse で利用することができます。

デバッガーのパースペクティブに切り替えるには、メニューコマンドから「**ウィンドウ | パースペクティブを開く | その他**」を選択します。表示されるダイアログ(以下のスクリーンショット)にて、デバッガーを選択(例えば、XSLT/XQをデバッグ)し、「**OK**」をクリックします。

入力ヘルパーまたはアクティブなドキュメントに対して選択されたデバッガーのパースペクティブが適用されます。この方法でユーザーはパースペクティブの切り替えを行うことができます。目的のパースペクティブを「**パースペクティブを開く**」サブメニュー(の「**その他**」アイテムの上)に表示させることで、パースペクティブにより素早くアクセスすることができます。この設定はカスタマイズダイアログにて行うことができます。

デバッガーを使用する方法については、このドキュメンテーション内の対応するセクションを参照ください:XSLT とXQuery、W SDL と SOAP。

(C) 2015-2021 Altova GmbH

24 メニューコマンド

ユーザーリファレンスには、XMLSpy のメニューコマンドに関する全てが記述されており、それらの一般的な使用方法について説明します。網羅的な内容になるよう努力はしておりますが、ユーザーリファレンスで扱われていない事柄に関する質問などありましたら、Altova Web サイトのFAQ またはディスカッションフォーラムを参照ください。適切な答えが見つからない場合、Altova サポートセンター[1052]にお問い合わせください。

準的なWindows コマンド(開く、保存、切り取り、コピー、貼り付けなど)は、ファイル[742] と編集[762] メニューに収められています。これらのメニューにはXML やインターネットに関するコマンドも収められています。

24.1 ファイルメニュー

ファイルメニューには、一般的な Windows ソフトウェア製品で使用されるような 新規作成[742]、開く[746]、保存[752]、印刷[758]、印刷設定[760]、と終了[760] などのファイルの操作を行うためのコマンドが収められています。XMLSpy には、XML 固有およびアプリケーション固有のコマンドが含まれています。

24.1.1 新規作成

このセクション:

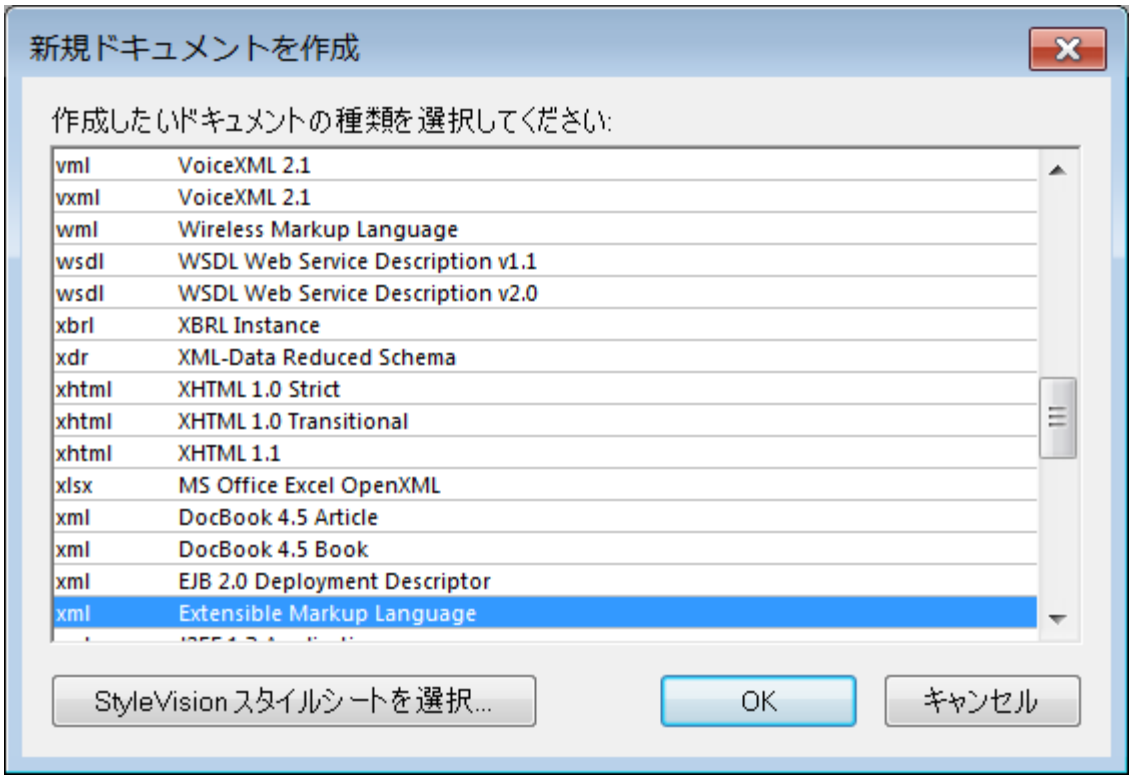
- アイコンとショートカット[742]
- 説明[743]
- 新規ドキュメントのテンプレートを作成[743]
- DTD/XML スキーマを新規 XML ドキュメントへ割り当て[744]
- 新規 XML ドキュメントのドキュメント要素を指定[745]
- 新規ドキュメントの作成時にSPS を割り当てる[745]

アイコンとショートカット

アイコン:	
ショートカット:	**Ctrl+N**

説明

新規作成コマンドは、新しいドキュメントを作成するために使用されます。**新規作成**をクリックすると、新規ドキュメントを作成ダイアログが表示され、作成するドキュメントの種類を選択することができます。作成したい種類のドキュメントが表示されていない場合は、XML を選択して、ファイルを保存する際に拡張子を変更してください。「ツール | オプション | ファイルの型 1018」セクションから新しい種類のファイルを追加することができます。

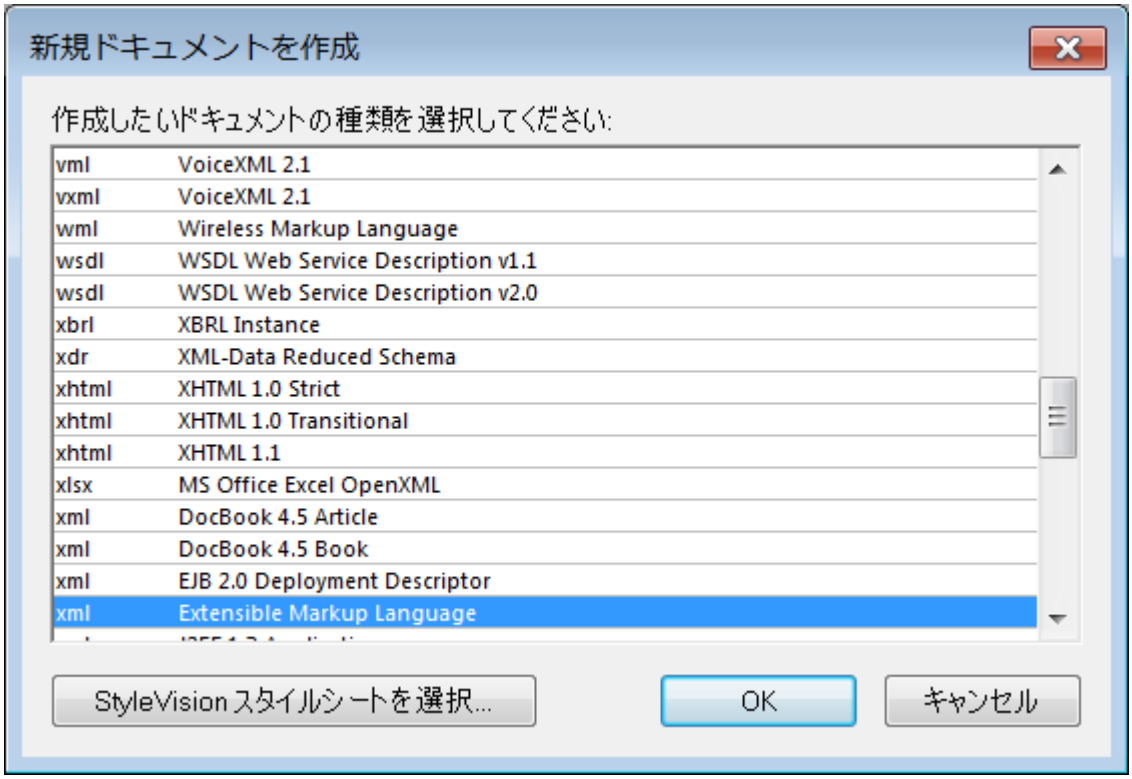


新規ドキュメントのテンプレート

様々な種類のファイルに対して複数のテンプレートを作成することができます。これらのテンプレートは、新規ドキュメントを作成ダイアログから直接開かれ、編集することができるようになります。新規ドキュメントを作成ダイアログに表示される独自のテンプレートを作成するには、まずテンプレートドキュメント作成し、全てのテンプレートが含まれるフォルダー内部にそのファイルを保存します。

以下の操作を行ってください:

1. Windows のエクスプローラーや、その他のナビゲーションツールを使用して、アプリケーション フォルダー 27 の `XMLSpy\Template` フォルダーを開きます。**new.xxx** という名前の基本的なファイルを選択します（.xxx というのが .xml や .xslt といったファイルの拡張子になります）。
2. そのファイルを XMLSpy で開き、必要な変更を行います。このファイルがテンプレートファイルになります。
3. 編集が終了したら、「**ファイル | 名前を付けて保存**」を選択し、ファイルを ¥Template フォルダーへ適当な名前とともに保存しなおします（ここでは例として「`MyXMLTemplate.xml`」という名前で保存しています）。これで「`MyXMLTemplate`」という名前のテンプレートが作成され、新規ドキュメントを作成ダイアログのリストに表示されるようになります。



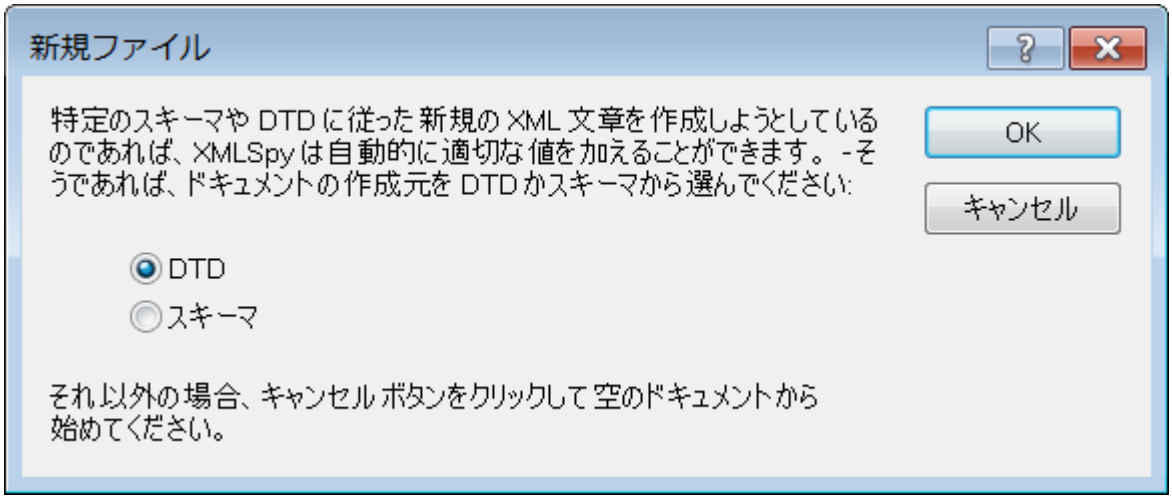
4. テンプレートを開くには、「**ファイル | 新規作成**」を選択し、そのテンプレート（この場合は「my-xml」）を選択します。

テンプレートを削除するには、該当するテンプレートファイルをテンプレートフォルダーから削除してください。

DTD/XML スキーマを新規 XML ドキュメントへ割り当て

標準的なスキーマ(DTD または XML スキーマ)をベースにした種類のドキュメントを新たに作成すると、適切な DTD または XML スキーマの割り当てとともにドキュメントは開かれます。例えば、XHTML ファイルは、DTD http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd が割り当てられた状態で開かれます。XML スキーマ(.xsd) ファイルは http://www.w3.org/2001/XMLschema スキーマドキュメントが割り当てられた状態で開かれます。

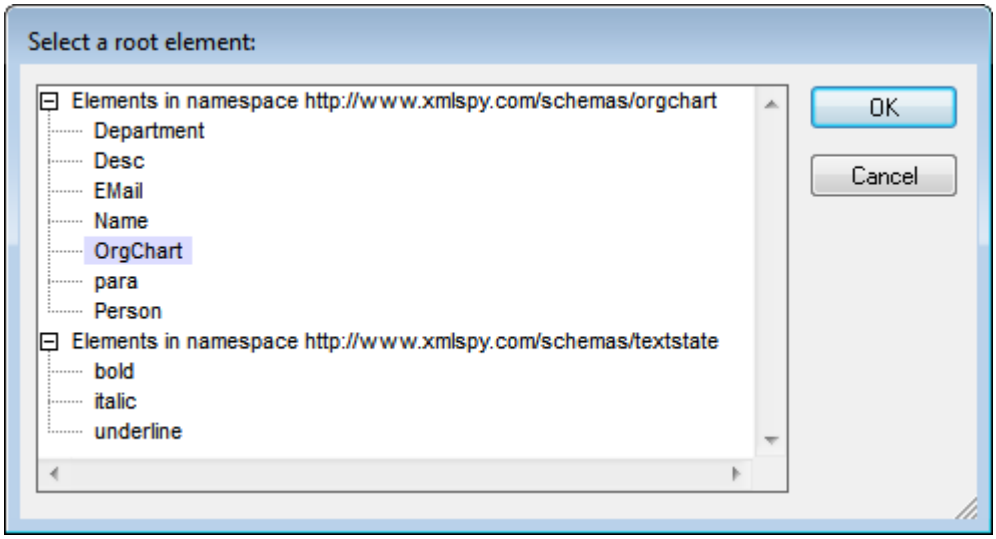
スキーマが定まっていないファイル(例えば空の XML ファイル)を新たに作成する場合、作成するドキュメントにスキーマ(DTD または XML スキーマ)を割り当てるよう促されます。



ドキュメントに対して DTD または XML スキーマの割り当てを行う場合は、「OK」をクリックすることでスキーマを参照することができます。このダイアログの「キャンセル」をクリックすると、どのスキーマも割り当てられていないファイルが作成されます。

新規 XML ドキュメントのドキュメント要素を指定

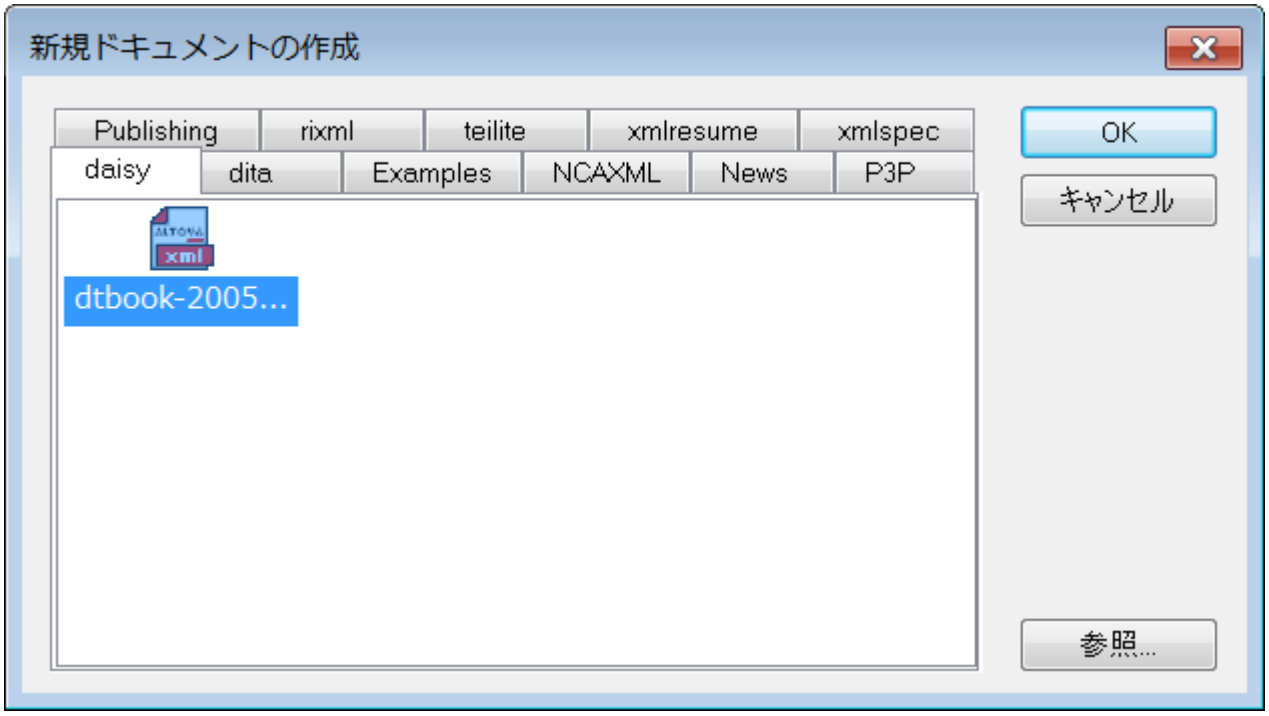
XML スキーマを選択する際に、そのスキーマ内にドキュメント（ルート）要素となることができるグローバル要素が複数個格納されている場合があります。新規ファイルダイアログにて選択された XML スキーマに、2つ以上のグローバル要素が定義されている場合、ルート要素を選択ダイアログが表示されます。



「OK」をクリックすると、この要素 (`OrgChart`) をルート要素とした XML ドキュメントが新たに作成されます。

新規ドキュメントの作成時に SPS を割り当てる

XML ドキュメントが新たに作成されると、ドキュメントを Authentic View で閲覧するために、StyleVision Power Stylesheet (`.sps` ファイル) を割り当てることができます。（上のスクリーンショットにある）新規ドキュメントを作成ダイアログにて、StyleVision スタイルシートを選択をクリックすると、新規ドキュメントの作成ダイアログが表示されます（以下のスクリーンショットを参照）。



フォルダータブから目的の SPS を参照することができます。他にも、参照ボタンをクリックして、目的のStyleVision Power スタイルシートを選択することができます。

24.1.2 開く

アイコンとショートカット

アイコン:	
ショートカット:	**Ctrl+O**

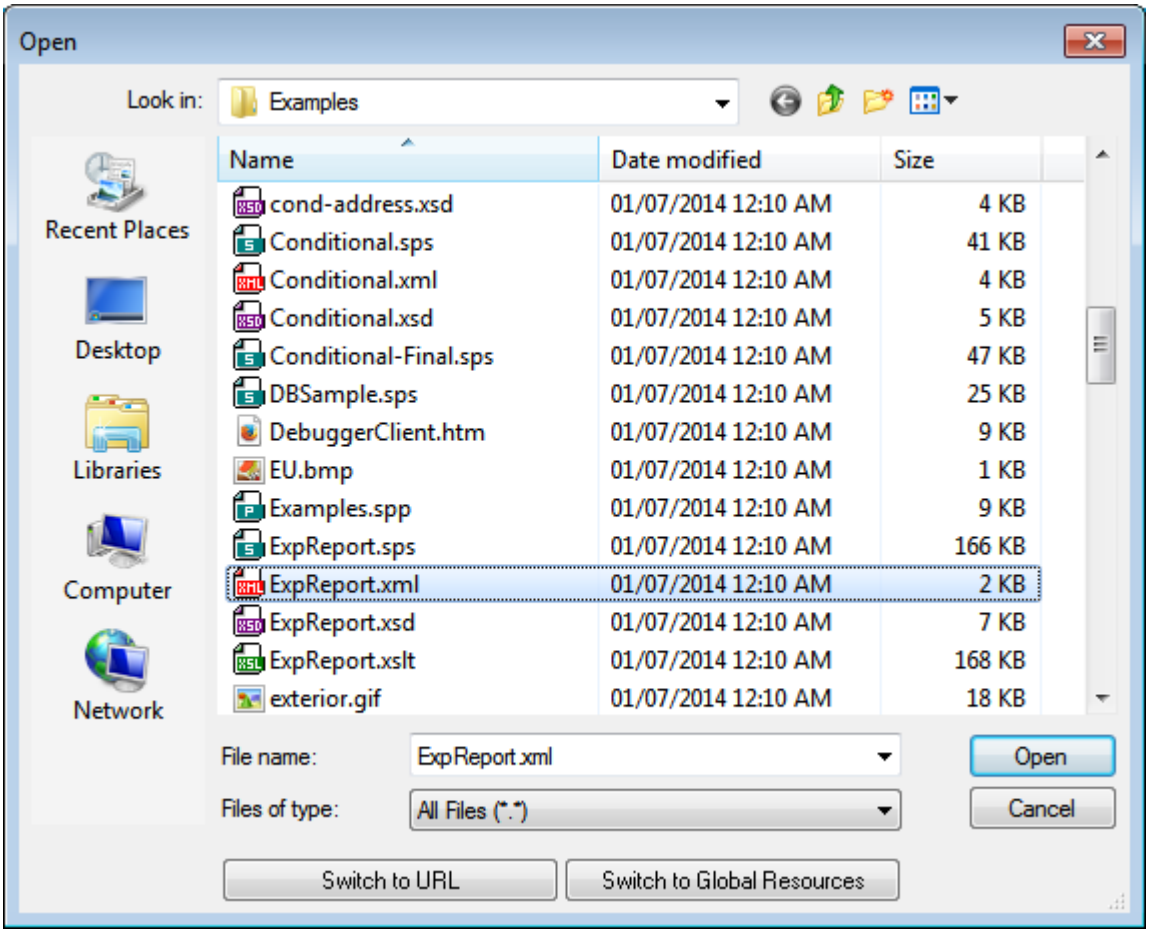
説明

開くコマンドにより、慣れ親しんだW indows の開くダイアログが表示され、XML に関するドキュメントや、テキストドキュメントを開くことができます。開くダイアログでは、2つ以上のファイルを開くことができます。ファイルの種類コンボボックスを使用することで、ダイアログボックスに表示されるファイルの型セクションを制限することができます(表示されるファイルの種類一覧は、オプションダイアログ(「ツール | オプション[1018]」)にて設定することができます)。XML ファイルが開かれると、整形式のチェックが行われます。ファイルが整形式で無い場合、整形式に関するエラーが表示されます。エラー箇所を修正して、メニューコマンドから XML | 整形式のチェック(F7)[839] を選択します。 ファイルを開く再に自動検証[1015] を行うよう選択しており、ファイルが妥当でない場合、エラーメッセージが表示されます。エラー箇所を修正して、メニューコマンドから「 XML | XML の検証 (F8)[841] 」を選択して、再度検証を行います。

- URL とグローバルリソースを使用してファイルを選択または保存する

  「ファイルを開く」および「ファイルの保存」ダイアログでは、URL または グローバルリソースを使用して必要なファイルを選択またはファイルを保存することができます(*下のスクリーンショット参照*)。選択プロセスに移動するために「URL に切り替える」または「グロ

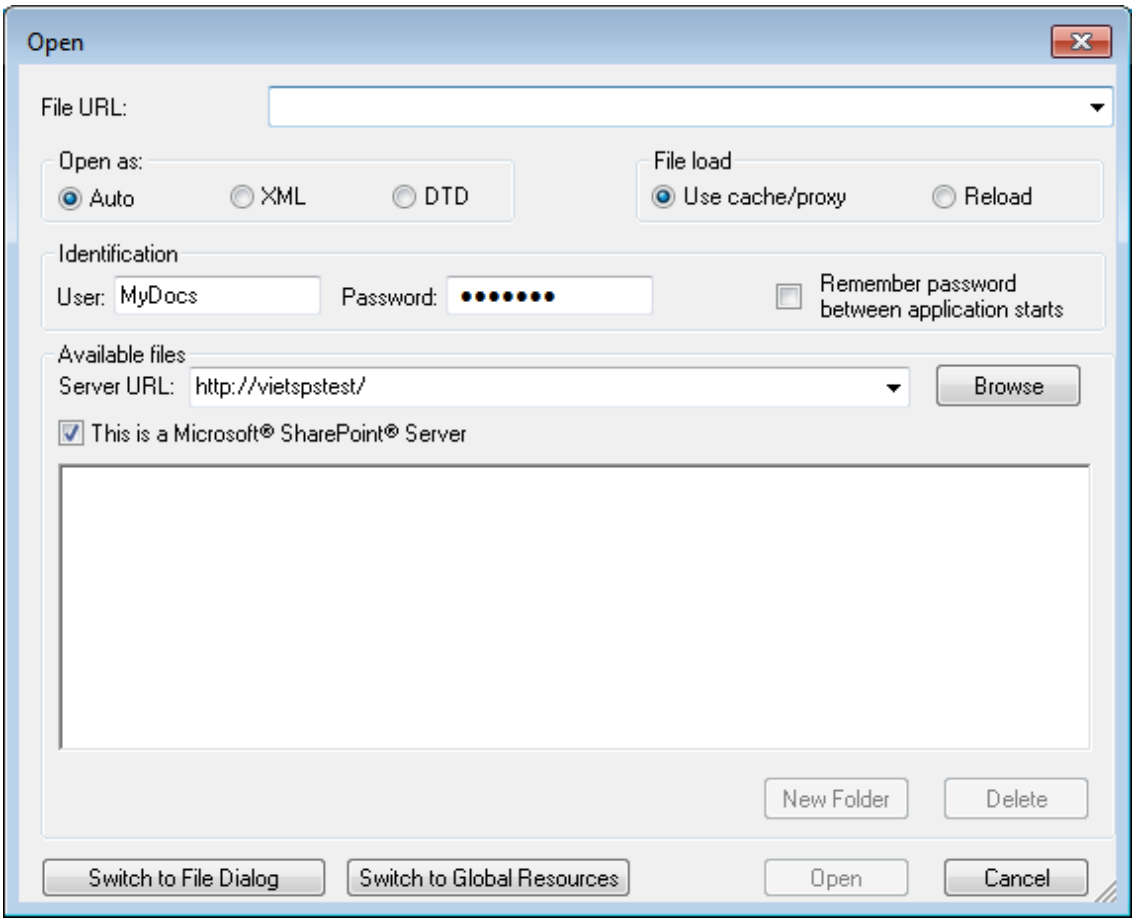
ーバルリソース」をクリックしてください。



URL を使用してファイルを切り替える

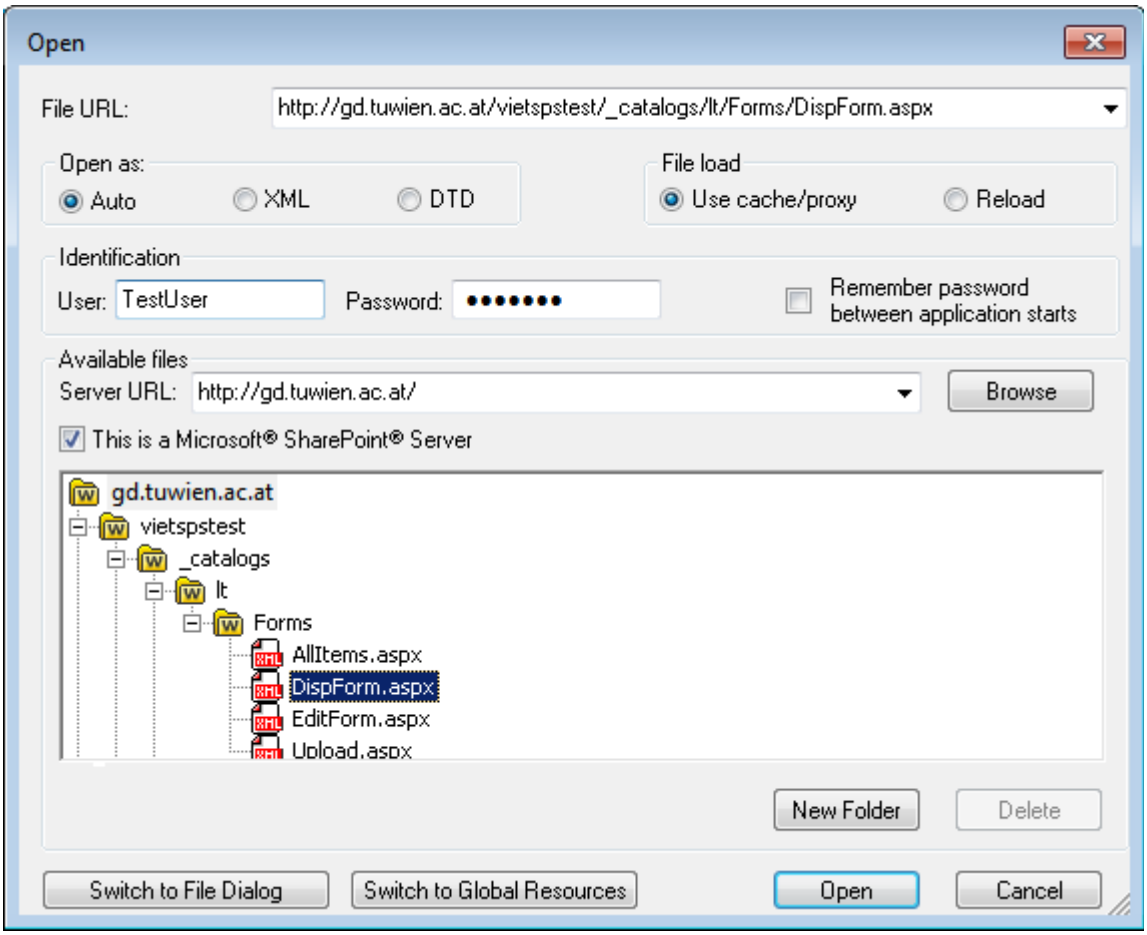
URL から(開くまたは保存する)ファイルを選択するには以下を行ってください:

1. 「URL に切り替える」コマンドをクリックします。これにより、開くまたは保存ダイアログの URL モードにスイッチされます。(*下のスクリーンショットは開くダイアログを表示しています*)。



2. アクセスするURLをサーバーURLフィールドに入力します（上のスクリーンショット）。サーバーがMicrosoft® SharePoint® Server の場合 *Microsoft® SharePoint® Server* チェックボックスをチェックしてください。この種類のサーバー上にあるファイルと作業する場合、下のMicrosoft® SharePoint® Server メモを参照してください。
3. サーバーがパスワードにより保護されている場合、ユーザーとパスワードフィールドに入力してください。
4. 「参照」をクリックして、サーバーのディレクトリ構造をナビゲートします。
5. フォルダーツリー内でロードするファイルを参照してクリックします。

(C) 2015-2021 Altova GmbH



ファイルURL がファイルURL フィールドに表示されます（*上のスクリーンショット参照*）。「開く」または「保存」ボタンを使用できるようになります。

6. 「開く」をクリックしてファイルをロードするか、「保存」をクリックして保存します。

以下の点に注意してください：

- W ebDAV をサポートするサーバーとMicrosoft SharePoint Servers のみで参照機能を使用することができます。サポートされるプロトコールはFTP、HTTP およびHTTPS です。
- ファイルを開く際のロード処理を更に管理するには、ローカルのキャッシュまたは（ファイルが以前にロードされている場合スピードを向上する）プロキシのサーバーからファイルをロードします。（ファイルが事前にロードされることにより更に処理スピードを向上することができます）。また、電子出版またはデータベースシステムなどの作業中のファイルを再ロードして、「再ロード」オプションを選択します

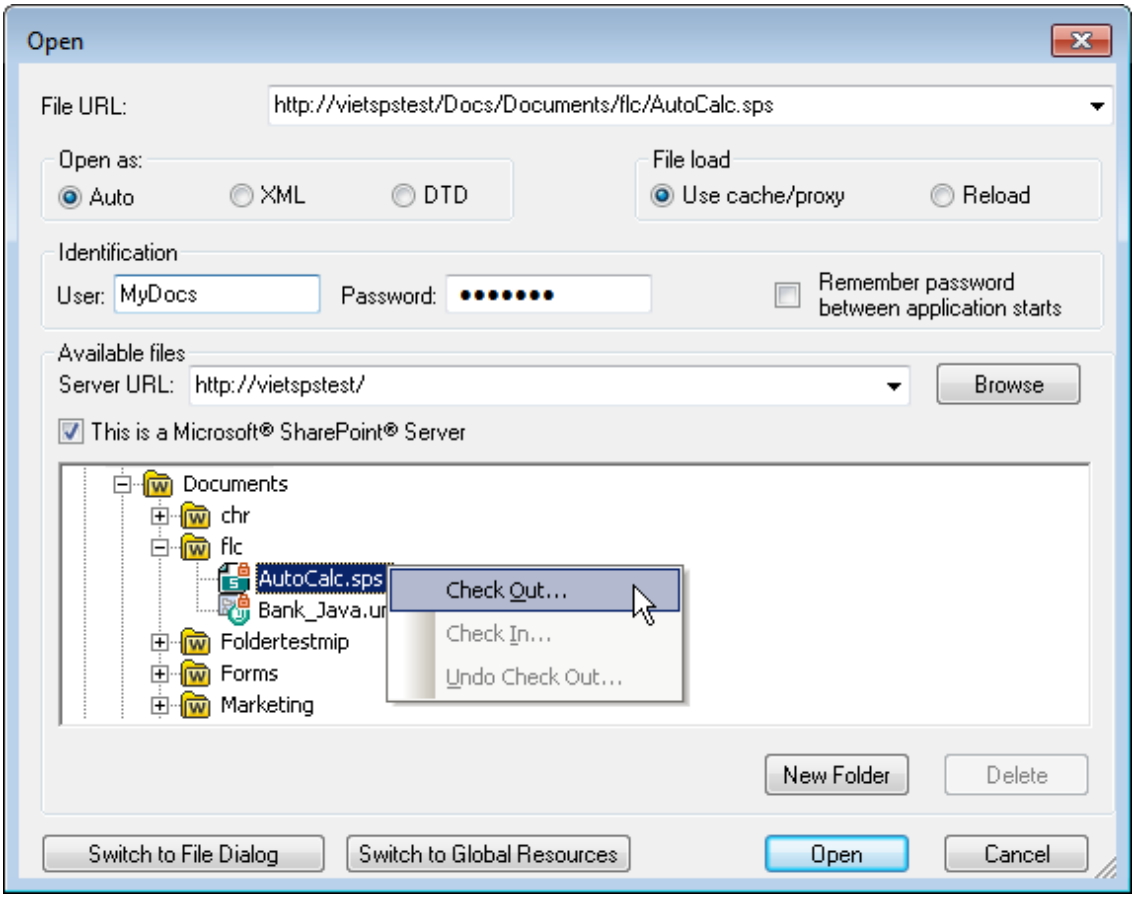
。

▼ Microsoft® SharePoint® Server メモ

Microsoft® SharePoint® Servers に関する以下の点に注意してください：

- 「利用可能なファイル」ペインに表示されるディレクトリ構造では、ファイルアイコンにはファイルのチェックイン/チェックアウトの状

態を表示するシンボルがあります（*下のスクリーンショット*）。



ファイルを右クリックすると、ファイルで使用することのできるコンテキストメニューがポップアップします（*上のスクリーンショット*）。

- 異なるファイルアイコンが以下に表示されています:

	チェックイン状態。
	チェックアウト可能。他のユーザーによりチェックアウト状態。
	チェックアウト不可能。ローカルに対してチェックアウト。

- ファイルをチェックアウトした後、Altova アプリケーションを使用して編集することができ、「ファイル | 保存（Ctrl+S）」を使用して保存することができます。
- 編集されたファイルを「URL を開く」ダイアログのコンテキストメニューからチェックインすることができます（*上のスクリーンショット参照*）または、アプリケーションのメインウィンドウのファイルタブを右クリックすることによりポップアップするコンテキストメニューを使用して（*下のスクリーンショット*）。

- ファイルが他のユーザーによりチェックアウトされている場合、チェックアウトすることができません。
- 他のユーザーによりファイルがチェックアウトされている場合、チェックアウトを行うことはできません。これによりファイルを変更す

ることなくサーバーに戻すことができます。

- 変更することなしにファイルがサーバーへ戻されます。ある Altova 製品でファイルをチェックアウトした場合、他の Altova 製品で同じファイルをチェックアウトすることはできません。この時点で使用できる Microsoft® SharePoint® Server をサポートする Altova アプリケーションのコマンドは、以下の通りです: 「**チェックイン**」および「**チェックアウトを元に戻す**」です。

▼ グローバルリソースを使用してファイルを選択または保存する

グローバルリソースを使用してファイルを開くまたは保存する場合、「**グローバルリソース**」をクリックします。グローバルリソースを選択するためのダイアログが表示されます。これらのダイアログは以下のセクションで説明されています: グローバルリソースの使用 670. グローバルリソースの説明に関してはこのドキュメントの グローバルリソース 660 セクションを参照してください。

24.1.3 再ロード

アイコン

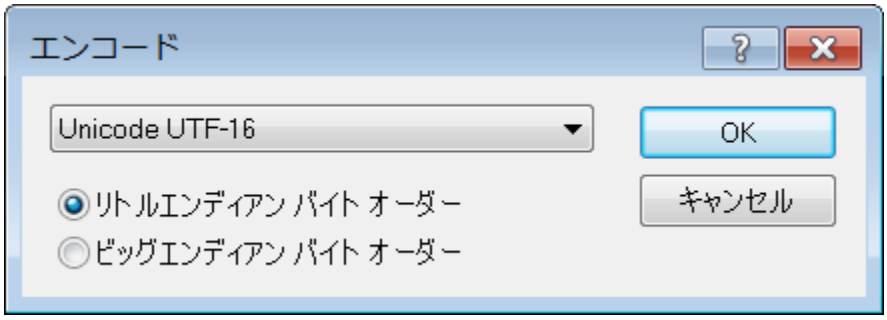
アイコン:	

説明

XMLSpy 外で変更された開かれているドキュメントを再ロードします。XMLSpy 外でドキュメントが変更されると、変更されたドキュメントの再ロードを問うプロンプトが表示されます。再ロードを選択すると、ファイルが最後に保存された時点からの変更が失われます。

24.1.4 エンコード

エンコードコマンドにより、(XML 以外も含む) 現在アクティブなドキュメントにて使用されているエンコーディングを確認し、他のエンコーディングを選択して、次回の保存時に反映させることができます。



XML ドキュメントにおいて、それまで使われていたものとは異なるエンコーディングを指定すると、XML 宣言内のエンコーディング指定もそれに合わせて変更されます。2 バイト文字ならびに 4 バイト文字エンコーディング (UTF-16、UCS-2、UCS-4) では、ファイルに使用されるバイトオーダーを指定することができます。XML ドキュメントのエンコーディングを変更するもう1つの方法は、ドキュメントの XML 宣言で使用されて

いるencoding 属性を直接編集することです。既存ならびに新規 XML ドキュメント、または非 XML ドキュメントのデフォルトエンコーディングは、オプションダイアログのエンコードセクション[1020]にて指定することができます。

メモ: ドキュメントを保存する際、XMLSpy はエンコーディングの指定を自動的にチェックして、ユーザーにより入力されたエンコーディングが認識できない場合は、ダイアログを表示します。更に、選択されたエンコーディングにより表現することのできない文字がドキュメントに含まれている場合、ファイルを保存する際に警告メッセージが表示されます。

24.1.5 閉じる、全て閉じる、非アクティブを全て閉じる

閉じる

閉じるコマンドにより、アクティブなドキュメントウィンドウが閉じられます。ファイルが変更されている場合(タイトルバーに表示されるファイル名の後にアスタリスク * が表示されます)、まずファイルを保存するか尋ねられます。

全て閉じる

全て閉じるコマンドにより、開かれているドキュメントウィンドウが全て閉じられます。ドキュメントのどれかが変更されている場合、まずファイルを保存するか尋ねられます。

非アクティブを全て閉じる

非アクティブを全て閉じるコマンドにより、アクティブなドキュメントウィンドウ以外の、全ウィンドウが閉じられます。ドキュメントのどれかが変更されている場合、まずファイルを保存するか尋ねられます。

24.1.6 保存、名前を付けて保存、全て保存

アイコンとショートカット

コマンド	*アイコン*	*ショートカット*
保存		**Ctrl+S**
名前を付けて保存		

保存

保存 (Ctrl+S) コマンドにより、アクティブなドキュメントのコンテンツを開かれた元のファイルに保存しなおします。ドキュメントを保存する際、整形式が自動的にチェック[839]されます。オプションダイアログ(ツール | オプション[1015])のファイルセクションにてオプションが指定されていれば、ファイルの検証も同時に行われます。XML 宣言のエンコード[1020]指定もチェックされ、ファイルの保存時にエンコーディングがドキュメントに対して適用されます。

名前をつけて保存

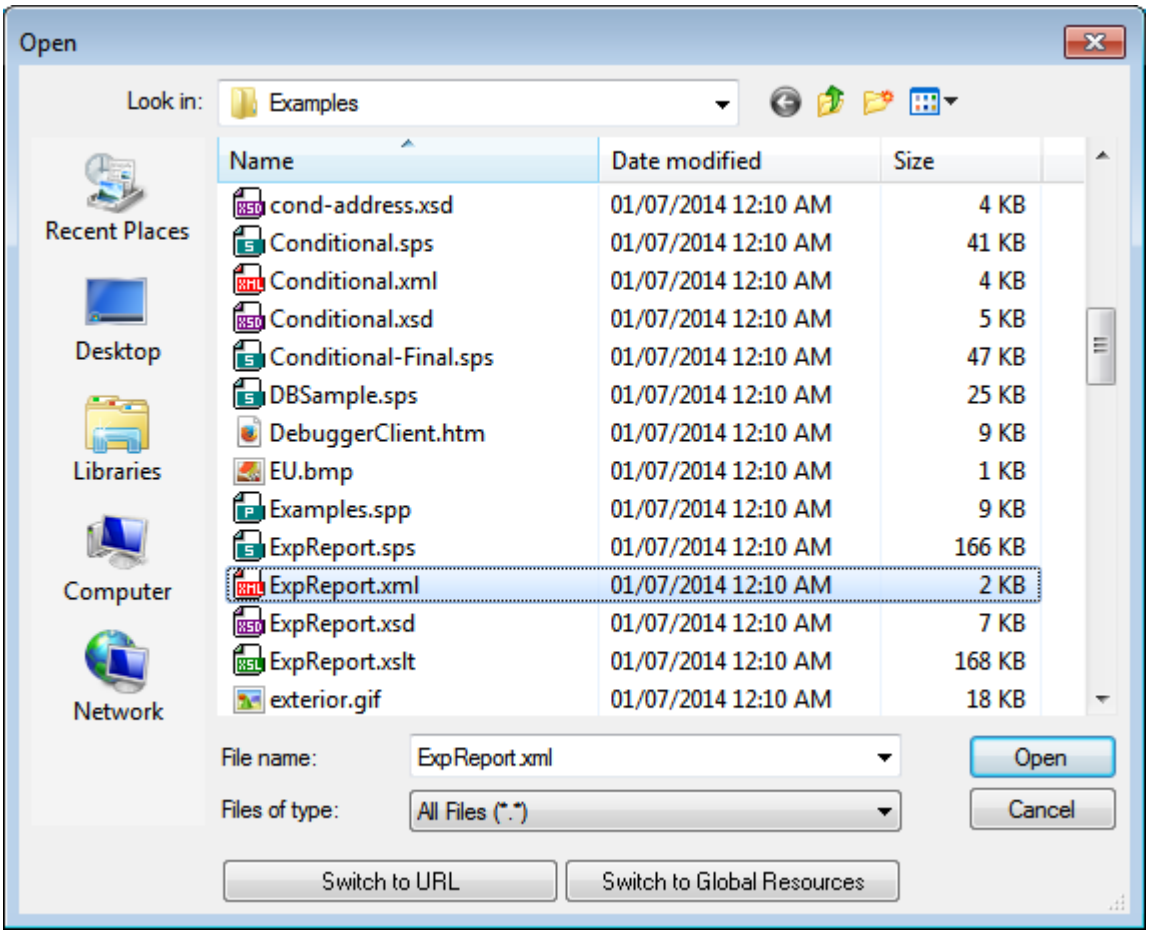
名前を付けて保存コマンドを選択すると、馴染みのある W indows 名前を付けて保存ダイアログボックスが表示され、アクティブなドキュメントの内容を保存するファイルの場所と名前を入力することになります。**保存**コマンドに対して行われるものと同じチェックがここでも行われます。

全て保存

全て保存 コマンドでは、開かれているドキュメントに対して行われた全ての変更を保存します。このコマンドは、複数のドキュメントを同時に保存する際に便利な機能になります。まだドキュメントが保存されていない場合（例えば、ファイルが新たに作成された後、まだ保存されていない場合）、そのドキュメントに対して名前を付けて保存ダイアログボックスが表示されます。

URL とグローバルリソースを使用してファイルを選択または保存する

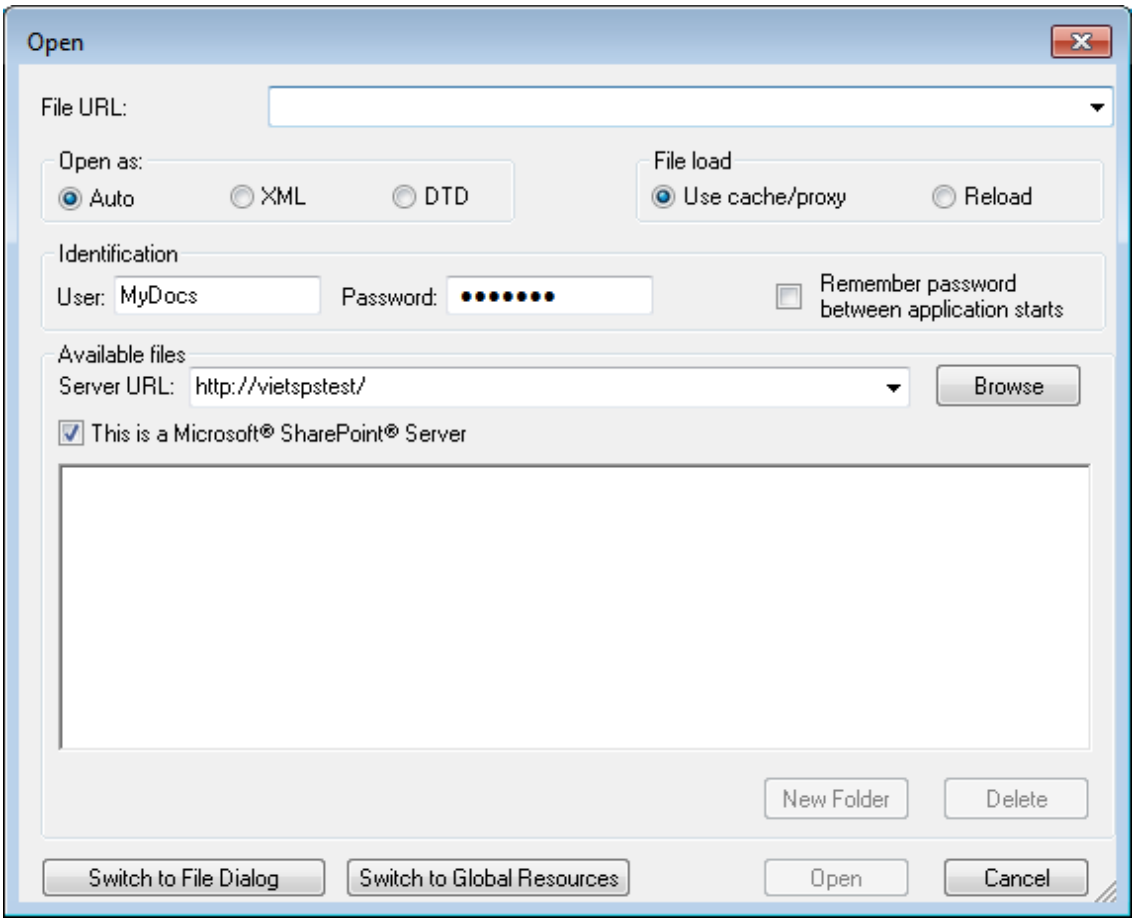
「ファイルを開く」および「ファイルの保存」ダイアログでは、URL または グローバルリソースを使用して必要なファイルを選択またはファイルを保存することができます（*下のスクリーンショット参照*）。選択プロセスに移動するために「**URL に切り替える**」または「**グローバルリソース**」をクリックしてください。



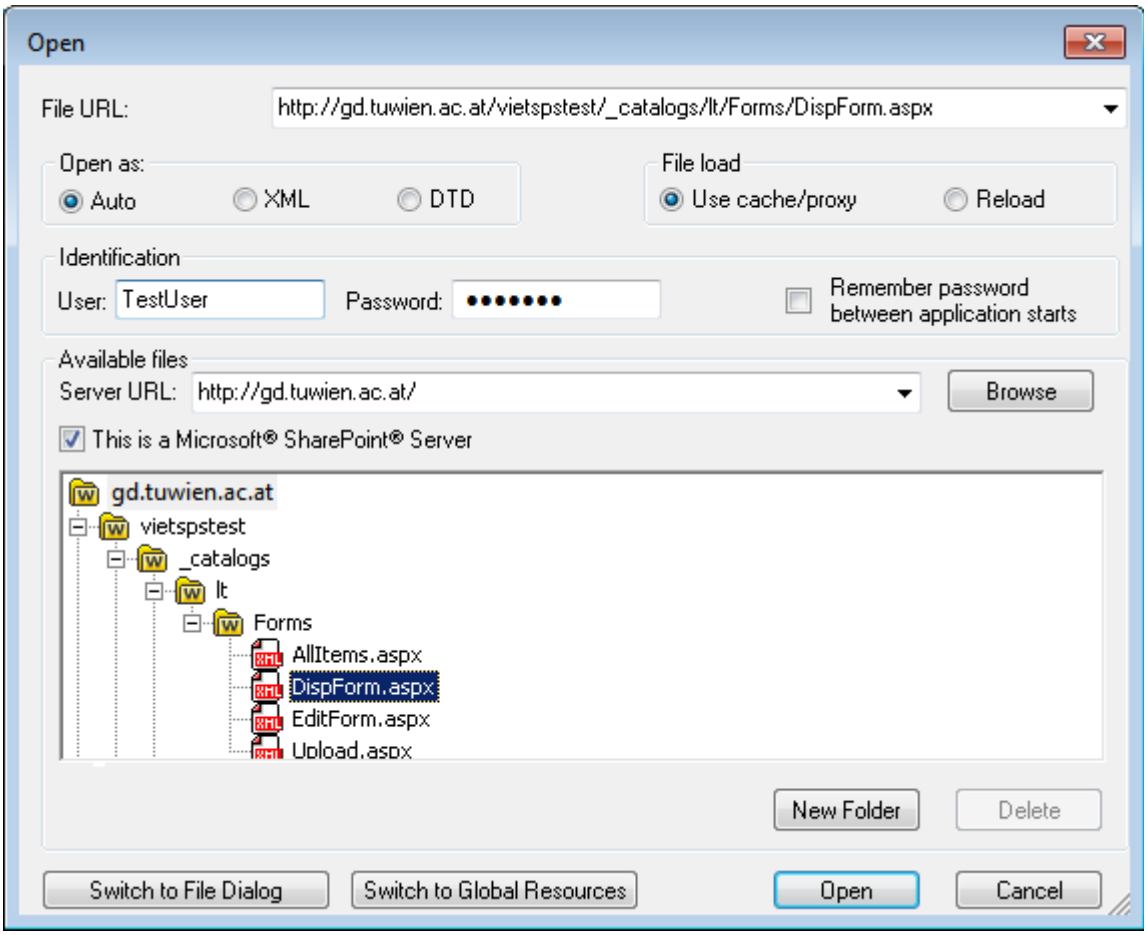
URL を使用してファイルを切り替える

URL から（開くまたは保存する）ファイルを選択するには以下を行ってください：

1. 「**URL に切り替える**」コマンドをクリックします。これにより、開くまたは保存ダイアログの URL モードにスイッチされます。（*下のスクリーンショットは開くダイアログを表示しています*）。



2. アクセスするURLをサーバーURLフィールドに入力します（上のスクリーンショット）。サーバーがMicrosoft® SharePoint® Server の場合 *Microsoft® SharePoint® Server* チェックボックスをチェックしてください。この種類のサーバー上にあるファイルと作業する場合、下のMicrosoft® SharePoint® Server メモを参照してください。
3. サーバーがパスワードにより保護されている場合、ユーザーとパスワードフィールドに入力してください。
4. 「参照」をクリックして、サーバーのディレクトリ構造をナビゲートします。
5. フォルダーツリー内でロードするファイルを参照してクリックします。



ファイルURL がファイルURL フィールドに表示されます（*上のスクリーンショット参照*）。「開く」または「保存」ボタンを使用できるようになります。

6. 「開く」をクリックしてファイルをロードするか、「保存」をクリックして保存します。

以下の点に注意してください：

- W ebDAV をサポートするサーバーとMicrosoft SharePoint Servers のみで参照機能を使用することができます。サポートされるプロトコールはFTP、HTTP およびHTTPS です。
- ファイルを開く際のロード処理を更に管理するには、ローカルのキャッシュまたは（ファイルが以前にロードされている場合スピードを向上する）プロキシのサーバーからファイルをロードします。（ファイルが事前にロードされることにより更に処理スピードを向上することができます）。また、電子出版またはデータベースシステムなどの作業中のファイルを再ロードして、「再ロード」オプションを選択します

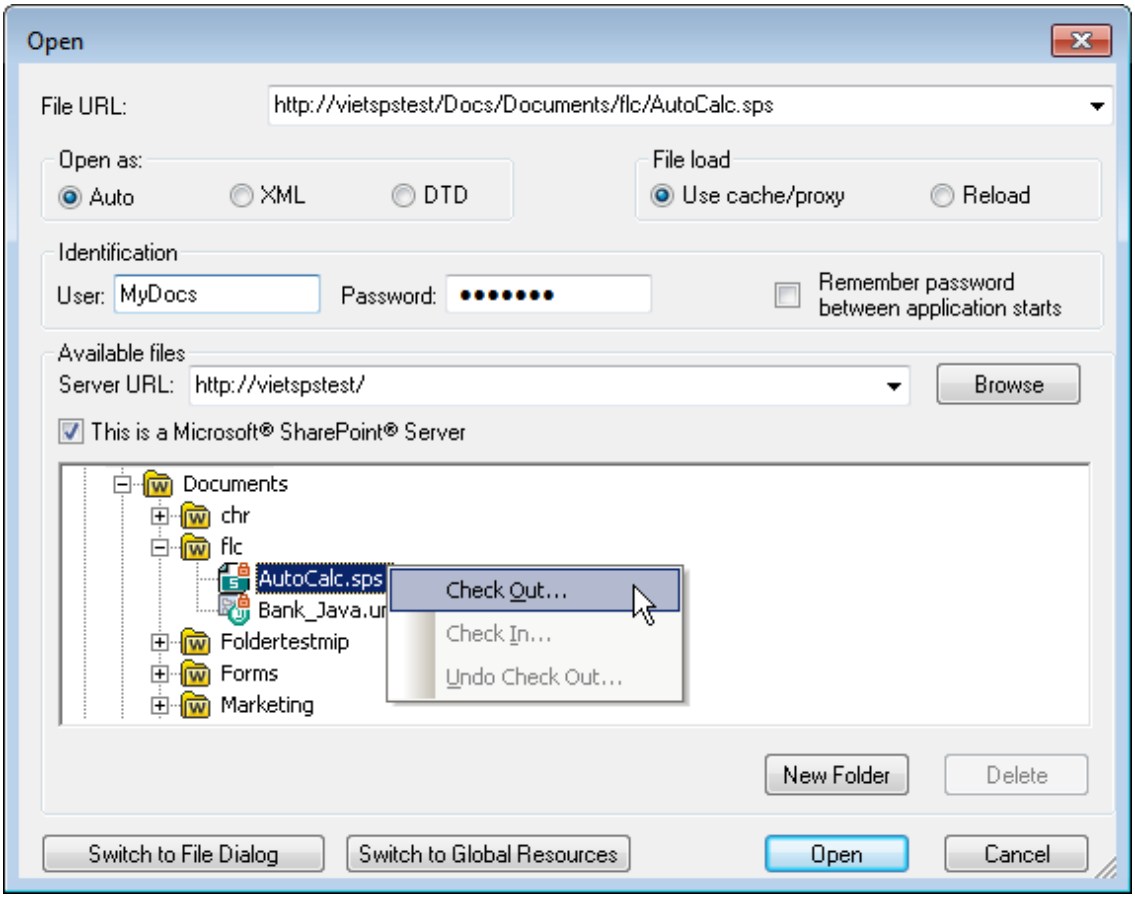
。

▼ Microsoft® SharePoint® Server メモ

Microsoft® SharePoint® Servers に関する以下の点に注意してください：

- 「利用可能なファイル」ペインに表示されるディレクトリ構造では、ファイルアイコンにはファイルのチェックイン/チェックアウトの状

態を表示するシンボルがあります（*下のスクリーンショット*）。



ファイルを右クリックすると、ファイルで使用することのできるコンテキストメニューがポップアップします（*上のスクリーンショット*）。

- 異なるファイルアイコンが以下に表示されています:

	チェック イン状態。
	チェック アウト可能。他のユーザーによりチェック アウト状態。
	チェック アウト不可能。ローカルに対してチェック アウト。

- ファイルをチェックアウトした後、Altova アプリケーションを使用して編集することができ、「ファイル | 保存（Ctrl+S）」を使用して保存することができます。
- 編集されたファイルを「URL を開く」ダイアログのコンテキストメニューからチェックインすることができます（*上のスクリーンショット参照*）または、アプリケーションのメインウィンドウのファイルタブを右クリックすることによりポップアップするコンテキストメニューを使用して（*下のスクリーンショット*）。

- ファイルが他のユーザーによりチェックアウトされている場合、チェックアウトすることができません。
- 他のユーザーによりファイルがチェック アウトされている場合、チェック アウトを行うことはできません。これによりファイルを変更す

ることなくサーバーに戻すことができます。

- 変更することなしにファイルがサーバーへ戻されます。ある Altova 製品でファイルをチェックアウトした場合、他の Altova 製品で同じファイルをチェックアウトすることはできません。この時点で使用できる Microsoft® SharePoint® Server をサポートする Altova アプリケーションのコマンドは、以下の通りです: 「**チェックイン**」および「**チェックアウトを元に戻す**」です。

▼ グローバルリソースを使用してファイルを選択または保存する

グローバルリソースを使用してファイルを開くまたは保存する場合、「**グローバルリソース**」をクリックします。グローバルリソースを選択するためのダイアログが表示されます。これらのダイアログは以下のセクションで説明されています: グローバルリソースの使用 670. グローバルリソースの説明に関してはこのドキュメントの グローバルリソース 660 セクションを参照してください。

24.1.7 メールで送信

アイコン

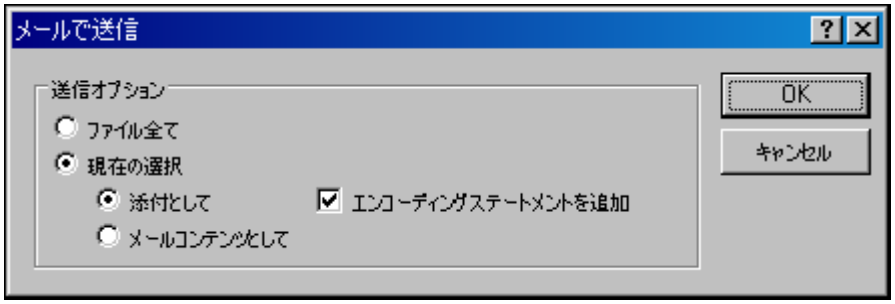
アイコン:	

説明

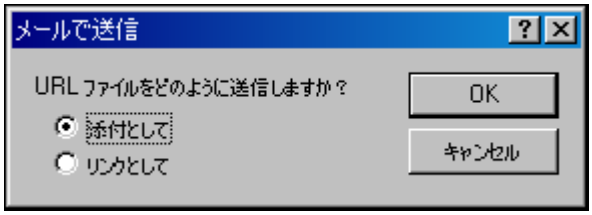
「**電子メールにて送信**」コマンドにより XML ドキュメント、または、XML ドキュメントからの選択を電子メールにて送信することができます。ドキュメント、または、ドキュメントの選択の種類により、ドキュメントを添付、コンテンツ、または、リンクとして送信することができます。詳細に関しては以下を参照してください。

送信できる内容	送信の方法
アクティブな XML ドキュメント	電子メールの添付
アクティブな XML ドキュメント内の選択	電子メールの添付または電子メールの内容として送信
プロジェクトウィンドウ内の 1 つまたは複数のファイル	電子メールの添付
プロジェクトウィンドウ内の 1 つまたは複数の URL	電子メールの添付またはリンクとして送信

「**電子メールにて送信**」コマンドがアクティブな XML ドキュメント内で呼び出されると、「電子メールにて送信」ダイアログがポップアップし、スクリーンショット内に送信のオプションが表示されます（*下のスクリーンショット*）。アクティブなファイル内でテキストが選択されておらず、「**電子メールにて送信**」コマンドが呼び出されると、*ファイル全てラジオボタンのみ*が有効化され、他のオプションは無効化されます（下のスクリーンショットを参照してください）。



プロジェクトウィンドウから送信されるファイルは、電子メールの添付のみで送信されるため、電子メールで送信ダイアログはスキップされ、選択されたファイルが添付された電子メールが開かれます。プロジェクトウィンドウ内のURL は、添付またはリンクとして送信することができます（*下のスクリーンショットを参照*）。URL を送信する方法を選択し、「OK」をクリックします。



24.1.8 印刷

アイコンとショートカット

アイコン:	
ショートカット:	**Ctrl+P**

説明

印刷コマンドにより、印刷ダイアログボックスが表示され、プリンターのオプションを指定することができます。現在アクティブなドキュメントを、表示されているままに印刷することができます。

グリッドビューにて**印刷**コマンドをクリックすると、印刷オプションダイアログが表示され（以下のスクリーンショットを参照）、XML ドキュメントの印刷オプションを指定することができます。このダイアログにある**印刷**をクリックすることで、プリンターオプションの印刷ダイアログが表示されます。

(C) 2015-2021 Altova GmbH

グリッドビューから印刷に利用することができるオプションは、以下のとおりです:

- 型ペインでは、出力に表示させるアイテムを選択します。
- 印刷範囲オプションでは、現在選択している箇所か、ファイル全体を印刷するかを指定します。
- 展開オプションにより、ドキュメントをそのまま印刷するのか、または子孫要素が全て展開された状態で印刷するのかを指定します。
- コンテンツオプションでは、ノード全体の印刷を行うのか、ノード名だけを印刷するのかを指定します。
- コンテンツ幅がページより広い場合ペインでは、コンテンツがページの幅に収まらない場合の処理を選択します。ページを分割オプションでは、コンテンツを(水平方向ならびに垂直方向で)ページごとに分割することで、ドキュメント全体を通常のサイズで印刷します。印刷されたページをつなぎあわせることで、ポスターを作成することができます。最初のページオプションでは、印刷領域の左端、最初のページのみが印刷されます。水平方向に収まらないコンテンツは印刷されません。このオプションは、グリッドビューに表示されている重要な情報の殆どが左側に表示されているような状態で利用することができます。水平に縮小オプションを使用すると、印刷される内容が水平方向に収まるよう、大きさを(水平方向ならびに垂直方向に)縮小して印刷します。ドキュメントは複数ページになる場合もあります。両方縮小オプションを使うと、ドキュメントが1ページに収まるように縮小されます。
- 印刷ボタンをクリックすることで、選択されたオプションによる印刷が行われます。
- プレビューボタンにより、印刷プレビューウィンドウが表示され、印刷を実際に行う前に、最終的に印刷される内容を確認することができます。
- 印刷設定ボタンをクリックすると、プリンターの設定ダイアログが表示され、この印刷ジョブだけに対しての用紙サイズや印刷の向き、その他のプリンターオプションを設定することができます。印刷設定 760 コマンドに関する記述も参照ください。

メモ:　グリッドビューのカラム幅を変更することで、印刷出力を最適化することができます。

プログラムロゴ

ライセンスを購入されている場合、プログラムロゴ、著作権に関する情報、そして登録に関する詳細を XMLSpy にて印刷されないように指定することができます。オプションダイアログのビューセクション 1025 にてこのオプションをご利用になれます。

24.1.9 印刷プレビュー、印刷設定

印刷プレビュー

テキストビュー、Authentic View、とブラウザービュー内の**印刷プレビュー**コマンドをクリックすると、ボタンをクリックすると、現在アクティブなドキュメントの印刷プレビューが表示されます。グリッドビューとスキーマビューから印刷オプションを選択することのできる印刷ダイアログボックスが開かれ、印刷プレビューを表示するために**プレビュー**ボタンをクリックします。

拡大や**縮小**ボタンを使用することで、プレビューの表示を最適化することができます。ページ全体が表示されている状態であれば、**1ページ / 2ページ**ボタンにより、同時に表示されるページの数を切り替えることができます。**次のページ**ならびに**前のページ**ボタンにより、ページ間の移動を行うことができます。ツールバーには前ページの印刷やプレビューウィンドウを閉じるオプションも用意されています。

メモ: 印刷プレビューにて背景色やイメージを有効にするには、(i) Internet Explorer の**ツール**メニューにて、**インターネットオプション**を選択し、詳細設定タブをクリックし、(ii) 設定以下にある印刷にて、*背景の色とイメージを印刷する* チェックボックスにチェックを入れ、(iii)「**OK**」をクリックします。

印刷設定

印刷設定コマンドでは、プリンターに関する印刷設定ダイアログボックスが表示され、用紙のフォーマットや印刷の向きといった設定を行うことができます。これらの設定は、その後行われる印刷ジョブにも引き継がれます。

24.1.10 最近使用されたファイル、終了

最近使用されたファイル

「**ファイル**」メニューの一番下に最近使用した順に9 つのファイルのリストが表示されます。名前をクリックしてファイルを開くことができます。キーボードを使用してファイルを開く場合、「**Alt+F**」を押して「**ファイル**」メニューを開き、開く番号を押します。

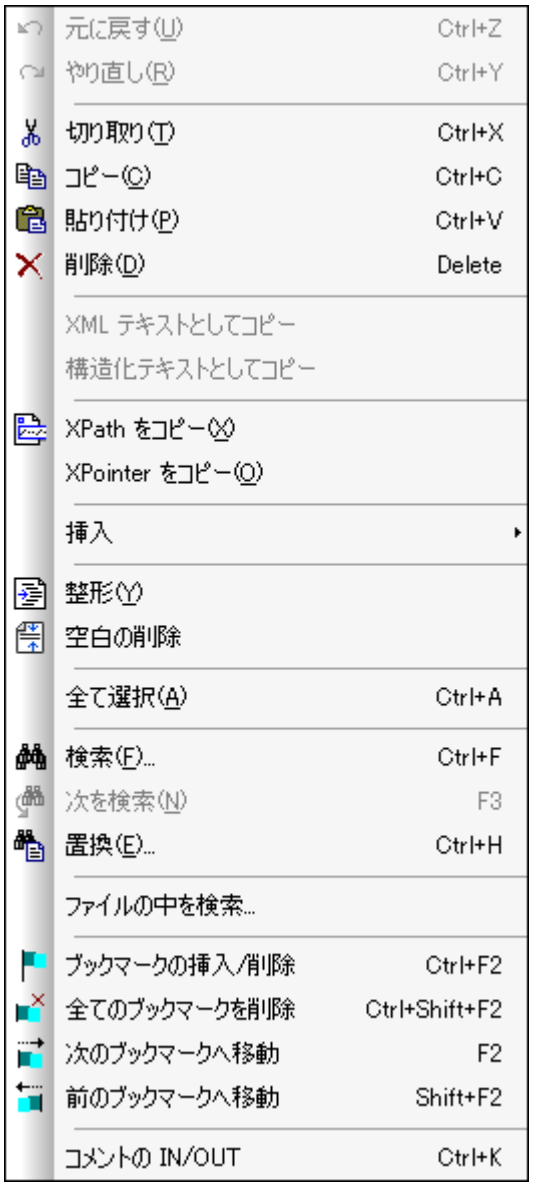
(C) 2015-2021 Altova GmbH

終了

XMLSpy を終了します。保存されていない変更のある開かれているファイルがある場合、これらの変更を保存するようにプロンプトされます。XMLSpy はプログラム設定への変更と最近使用されたファイルの情報も保存します。

24.2 編集メニュー

編集メニューには、XMLSpy にて開かれているドキュメントの編集を行うためのコマンドが用意されています。標準的な 元に戻す[763]、やり直し[763]、切り取り[763]、コピー[763]、貼り付け[763]、削除[763]、全て選択[772]、検索[773]、次を検索[773] および 置換[776] コマンドなどが用意されています。



XMLSpy では以下の様な特別なコマンドが用意されています:

- 選択された箇所を XML テキストとしてコピー[764],
- 構造化されたテキストとしてコピー[764]
- 選択されたアイテムの XPath セレクターをクリップボードにコピー[768]
- ブックマークの挿入や削除やブックマークのナビゲーション

24.2.1 元に戻す、やり直し

アイコンとショートカット

コマンド	*アイコン*	*ショートカット*
元に戻す		**Ctrl+Z**
やり直し		**Ctrl+Y**

元に戻す

元に戻す コマンド では、回数に制限の無いアンドゥ(作業の取り消し)がサポートされます。この機能を使って行われた全ての操作を取り消すことができます。元に戻すコマンドの記録は、ドキュメントが保存されても蓄えられ、変更を保存する前の状態まで操作を取り消すことができます。**元に戻す** と **やり直し** コマンドを使用して履歴を戻り、また、先に進むことができます(*やり直し コマンドを参照してください*)。

やり直し

やり直し コマンド では、それまでに取り消されたコマンドをやり直し、それまで行われた作業の記録を辿ることができます。**元に戻す**や、**やり直し**コマンドを使用することで、これまでの記録を行き来することができます。

24.2.2 切り取り、コピー、貼り付け、削除

アイコンとショートカット

コマンド	*アイコン*	*ショートカット*
切り取り		**Ctrl+X** または **Shift+Del**
コピー		**Ctrl+C**
貼り付け		**Ctrl+V**
削除		**Del**

切り取り

切り取り コマンド は、選択されたテキストまたはイメージをクリップボードにコピーして、オリジナルの場所にあるオブジェクトを削除します。

コピー

コピー コマンド では、選択されたテキストやアイテムをクリップボードにコピーします。このコマンドを使用することで、XMLSpy 内部でデータの複製を行い、他のアプリケーションへデータを移動することができます。

メモ グリッドビューからテキスト形式の要素をクリップボードにコピーするには XML テキストとしてコピー[764] と構造化テキストとしてコピー[764] するという、2つの方法があります。「オプション | ツール」ダイアログの編集タブ[1021] により、コピーコマンドが行われた際に、どちらの処理が行われるかを選択することができます。

貼り付け

貼り付け コマンド により、クリップボードにコピーされたコンテンツを、現在カーソルがある位置に挿入することができます。

削除

削除 コマンド を使うことで、現在選択されているテキスト やアイテムを、クリップボードにコピーすること無く削除することができます。

24.2.3 XML テキストとしてコピー

XML テキスト としてコピーコマンドを使用すると、XML ソース XML テキストを扱うことができる外部アプリケーションに、簡単にデータを受け渡すことができます。グリッドビューにてドキュメントの編集を行なっている場合、表示されている要素を XML テキスト形式でクリップボードにコピーすることができます。

```
<row>
    <para align="left">
        <bold>Check the FAQ</bold>
    </para>
    <para>
        <link mode="internal">
            <link_section>support</link_section>
            <link_subsection>faq30</link_subsection>
            <link_text>XMLSPY 4.0 FAQ</link_text>
        </link>
        <link mode="internal">
            <link_section>support</link_section>
            <link_subsection>faq25</link_subsection>
            <link_text>XMLSPY 3.5 FAQ</link_text>
        </link>
    </para>
</row>
```

XML テキスト としてコピーコマンドでは、現在アクティブな設定を使用して、コピーの対象となるテキストが自動的に整形されます。これらの設定は、オプションダイアログ(「**ツール | オプション**」)のファイルセクションにあるファイルを保存セクションにて変更することができます。グリッドビューからは、構造化テキスト としてコピー 764 コマンドを使用して、テーブルのグリッドビューの表示から XML データをコピーすることができます。

24.2.4 構造化テキストとしてコピー

構造化テキスト としてコピーコマンドは、要素を表示されているままにクリップボードへコピーします。このコマンドは、グリッドビューからテーブル形式のデータをコピーするのに便利な機能となります。

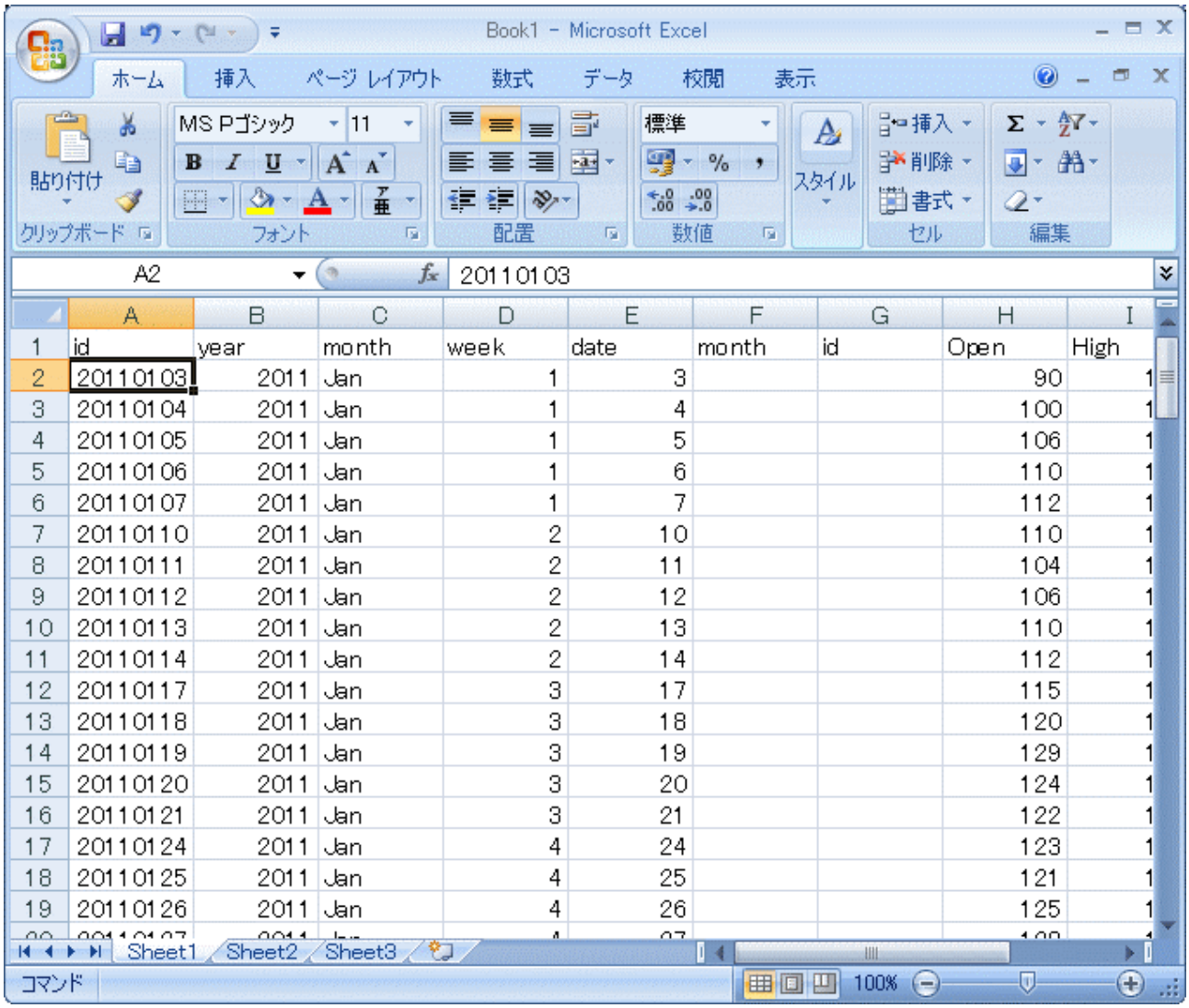
- テーブルビューから個別の行をコピーする 764
- テーブルビューからテーブル全体をコピーする 766
- 標準グリッドビューからコピーする 766

テーブルビューから個別の行をコピーする

下のスクリーンショット内では、グリッドビューの**テーブルビュー** 内の2 つの行が構成されたテキスト としてコピーされます。

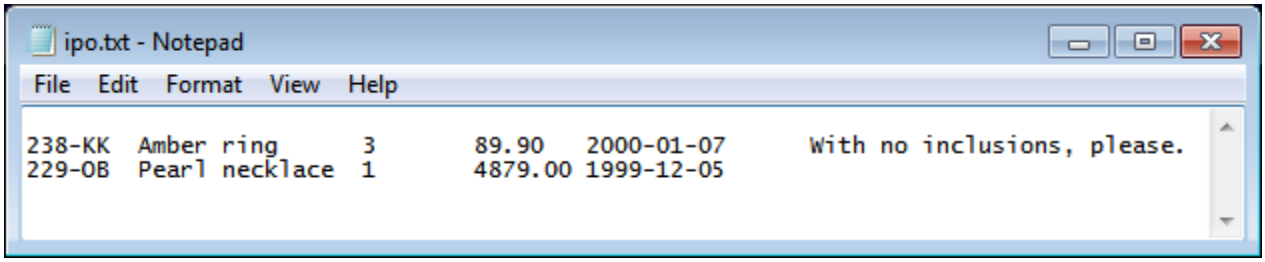
expense-item (3)	type	expto	Date	expense	description
1	Lodging	Sales	2003-01-01	122.11	
2	Lodging	Development	2003-01-02	122.12	Played penny arcade
3	Lodging	Marketing	2003-01-02	299.45	Treated Clients

次の2つの下のスクリーンショットは、グリッドビュー内の構成されたテキストとしてコピーされたデータを表示しています。上のスクリーンショットは、Microsoft Excel ドキュメントとNotepad ドキュメント内に貼り付けられたデータを表示しています。



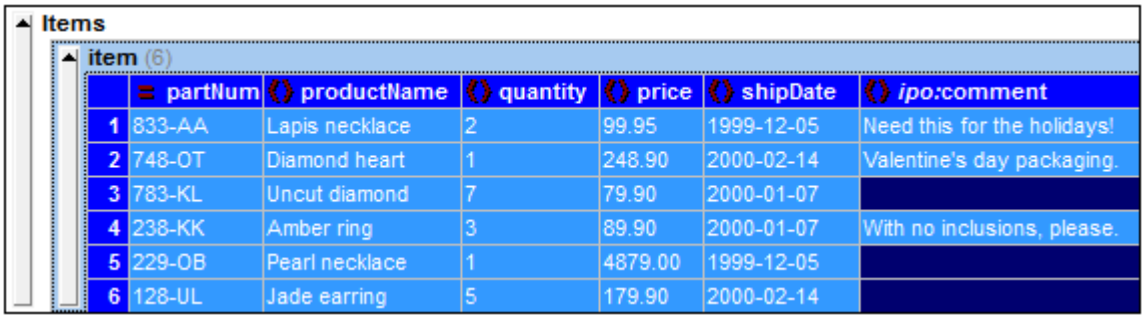
	A	B	C	D	E	F	G	H	I
1	id	year	month	week	date	month	id	Open	High
2	20110103	2011	Jan	1	3			90	1
3	20110104	2011	Jan	1	4			100	1
4	20110105	2011	Jan	1	5			106	1
5	20110106	2011	Jan	1	6			110	1
6	20110107	2011	Jan	1	7			112	1
7	20110110	2011	Jan	2	10			110	1
8	20110111	2011	Jan	2	11			104	1
9	20110112	2011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

Excel は自動的にセルテキストのそれぞれを、テキストの構文フォームをベースにフォーマットしますが、(*上のスクリーンショット*) Notepad はセルテキストを文字列として貼り付けます(*下のスクリーンショット*)。



テーブルビューからテーブル全体をコピーする

下のスクリーンショットでは、(`item (6)` ノードエントリをクリックすることにより)テーブル全体が選択されています。



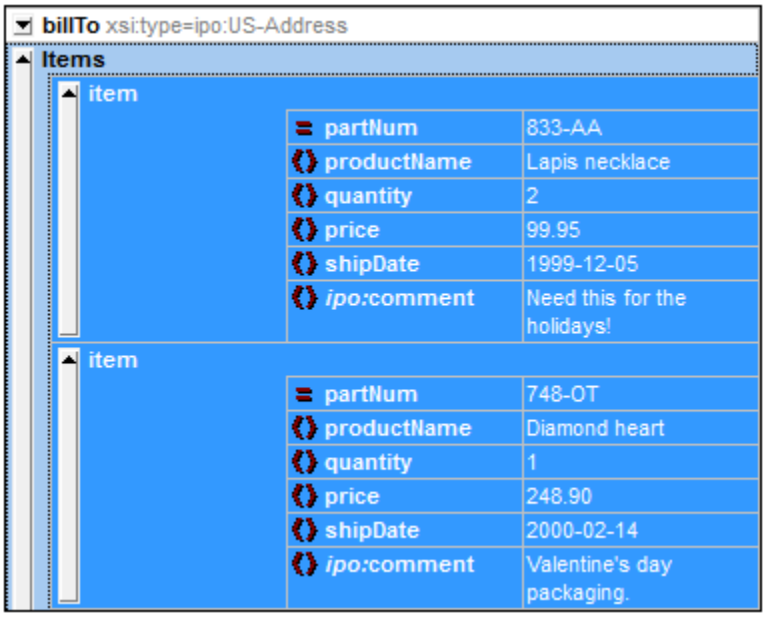
	partNum	productName	quantity	price	shipDate	ipo:comment
1	833-AA	Lapis necklace	2	99.95	1999-12-05	Need this for the holidays!
2	748-OT	Diamond heart	1	248.90	2000-02-14	Valentine's day packaging.
3	783-KL	Uncut diamond	7	79.90	2000-01-07	
4	238-KK	Amber ring	3	89.90	2000-01-07	With no inclusions, please.
5	229-OB	Pearl necklace	1	4879.00	1999-12-05	
6	128-UL	Jade earring	5	179.90	2000-02-14	

テーブルの列のタイトル以外の個別の行のための結果がコピーされます。下のスクリーンショットは、Microsoft Excel ワークシートにコピーして貼り付けられたデータを表示しています。XML ドキュメント内では、列ヘッダーは、要素または属性名であることに注意してください。

	A	B	C	D	E	F
1	partNum	productName	quantity	price	shipDate	ipo:comment
2	833-AA	Lapis necklace	2	99.95	12/05/1999	Need this for the holidays!
3	748-OT	Diamond heart	1	248.9	02/14/2000	Valentine's day packaging.
4	783-KL	Uncut diamond	7	79.9	01/07/2000	
5	238-KK	Amber ring	3	89.9	01/07/2000	With no inclusions, please.
6	229-OB	Pearl necklace	1	4879	12/05/1999	
7	128-UL	Jade earring	5	179.9	02/14/2000	

標準グリッドビューからコピーする

上に表示されている例はグリッドビューのテーブルビュー(グリッドビュー内でテーブルビューがオンになっている状態)からデータをコピーしています。標準のグリッドビューでは、(テーブルビューがオフに切り替えられている状態)、**構造化テキストとしてコピー** は、グリッドビューで表示されているとおり、選択されたデータをコピーします。しかしながら、構造は、テーブルではありません。例えば、下のスクリーンショットで通常のグリッドビュー内の2つの `item` 子を持つsアイテムノードを表示しています。



Microsoft Excel 内にコピーされたテキストが貼り付けられると、ワークシートは以下のスクリーンショットのようになります。

	A	B	C	D
1	Items			
2		item		
3			partNum	833-AA
4			productName	Lapis necklace
5			quantity	2
6			price	99.95
7			shipDate	12/05/1999
8			ipo:comment	Need this for the holidays!
9		item		
10			partNum	748-OT
11			productName	Diamond heart
12			quantity	1
13			price	248.9
14			shipDate	02/14/2000
15			ipo:comment	Valentine's day packaging.

XML データを構造化されたテキストとしてコピーするほか、構造化テキストとしてコピーコマンドを使用しても、データをXML テキストとしてグリッドビューからコピーすることができます 764。

24.2.5 XPath をコピー

XPath をコピーコマンドはテキストビューならびにグリッドビューで使用可能で、現在選択されているノードからXPath 条件式を作成し、クリップボードにコピーします。この機能により、条件式をドキュメント（例えばXSLT ドキュメント）に貼り付けることができます。全ての条件式は、ドキュメントルートから開始されます。は、特定の要素ではなく、その階層レベルにある全ての要素が選択されます。テキストビューの場合、その単一要素が選択されます。例えば、2番目のCompany 要素から見て3番目のPerson という要素以下にあるLastName という名前の要素を選択した場合、作成されるXPath 条件式は以下のようになります
: `/Companies/Company[2]/Person[3]/LastName`

メモ: グリッドビューでは、コンテキストメニューからも XPath をコピーコマンドへアクセスすることができます。

24.2.6 XPointer をコピー

XPointer をコピーコマンドは、テキストビューならびにグリッドビューで使用可能で、現在選択されているノードのelement() スキーマXPointer を作成し、クリップボードにコピーします。この機能により、XPointer をドキュメント（例えば、XML ドキュメント内部にあるXInclude 要素内のxpointer 属性）に貼り付けることができます。

XPointer のelement() スキーマからは、element(/1/3) という形式が返され、この例ではドキュメント要素（ルート要素）にある3番目の要素が選択されます。以下の点に注意してください:

- element() スキーマでは属性を表現することができません。XMLSpy で属性が選択されている場合、以下のことが起こります：グリッドビューでは、XPointer をコピーコマンドが無効になります。テキストビューでは、その属性の親となる要素を元に XPointer が生成されます。
- 複数の要素を選択することはできません。グリッドビューで複数の要素が選択されている場合、XPointer をコピーコマンドは無効になります。テキストビューでは、選択された複数要素の親となる要素を元に XPointer が生成されます。

メモ: グリッドビューでは、コンテキストメニューからも XPointer をコピーコマンドにアクセスすることができます。

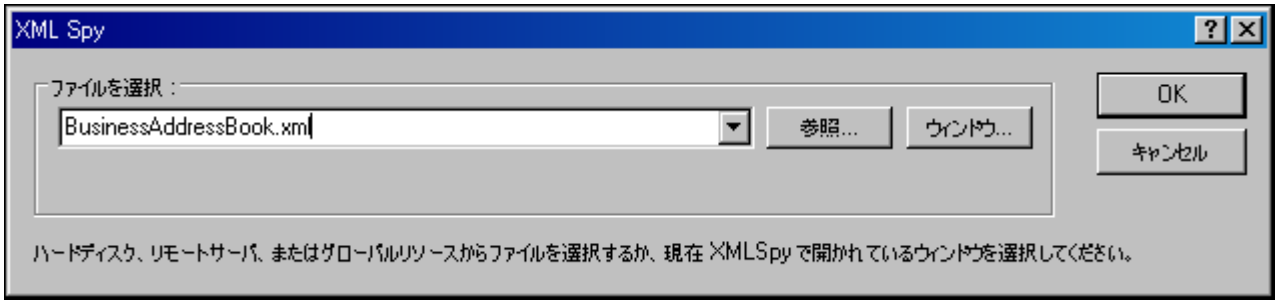
24.2.7 挿入

挿入コマンドを選択、またはマウスオーバーすると、以下に説明されるような3つのコマンドが表示されます:

- ファイルパスの挿入 768
- XInclude の挿入 769
- エンコードされた外部ファイルの挿入 771

ファイルパスの挿入

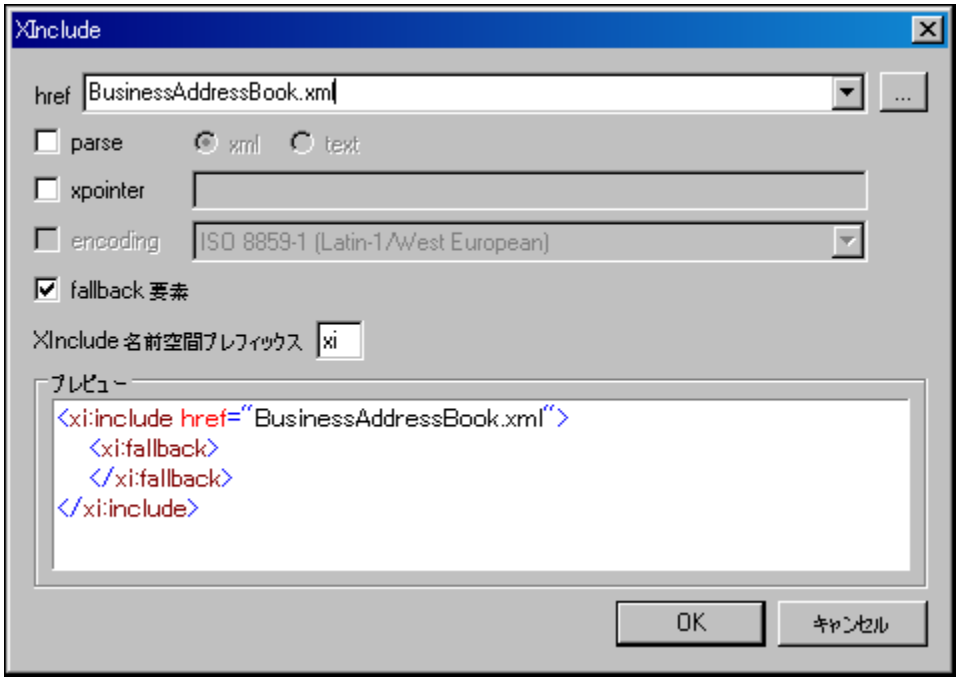
ファイルパスコマンドは、テキストビューまたはグリッドビューで表示されている、あらゆる種類のドキュメントで使用することができます。この機能を使用することで、カーソルがある位置にファイルのパスを挿入することができます。このコマンドをクリックすると、以下のスクリーンショットに示されるダイアログが表示され、目的のファイルを選択することができます。



目的のファイルは、以下にある方法から選択することができます：(i) ファイル、URL、またはグローバルリソースを参照（参照 746 ボタンを使用）、(ii) ウィンドウから既に開かれているファイルを選択（ウィンドウボタンを使用）。選択をが終わると、「OK」をクリックします。選択されたファイルのパスが、アクティブなドキュメントのカーソル位置に挿入されます。

XInclude の挿入

XInclude コマンドは、テキストビューまたはグリッドビューで使用することができ、新たな XInclude 要素を、テキストビューにてカーソルが選択している箇所、またはテキストビューやグリッドビューにて選択されているアイテムの直前に挿入することができます。グリッドビューにて選択されているのが属性の場合、XInclude 要素は、属性の後か、属性の親要素が持っている最初の子要素の前に挿入されます。このコマンドを選択すると、XInclude ダイアログが表示されます（以下のスクリーンショットを参照）。



インクルードされる XML ファイルを href テキストボックスに入力します（または、テキストボックスの右側にある… （参照）ボタンをクリックすることで、ファイルの参照を行うこともできます）。ファイル名が href 属性の値として、XML ドキュメント内に入力されます。XInclude (xi:include) 要素の parse、xpointer や encoding 属性、そして xi:include の fallback 子要素も、このダイアログにて挿入することができます。対応するチェックボックスにチェックを入れ、必要な値を選択/入力してください。fallback 要素の場合は、チェックボックスにチェックを入れることで、空要素を挿入することができます。fallback 要素のコンテンツは、編集ビューにて追加する必要があります。

parse 属性により、インクルードされるドキュメントが XML またはテキストとして解析（パース）されるかが決定されます（デフォルトでは XML

という値がセットされており、指定を行う必要はありません)。xpointer 属性は、href 属性により指定されるドキュメント内フラグメント(つまり、インクルードされるフラグメント)を特定します。encoding 属性ではインクルードされるドキュメントのエンコーディングが指定され、XMLSpy により、インクルードするドキュメントの内容がインクルード元のドキュメントエンコーディングに合う形で変換されます。インクルードされるドキュメントが見つからない場合は、xi:include 要素がfallback 子要素により置換えられます。

XInclude により2つのXML ドキュメントをインクルードするXML ドキュメントのサンプルを以下に示します:

```
<?xml version="1.0" encoding="UTF-16"?>
<AddressBook xsi:schemaLocation="http://www.altova.com/sv/myaddresses AddressBook.xsd"
       xmlns="http://www.altova.com/stylevision/tutorials/myaddresses"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:xi="http://www.w3.org/2001/XInclude">
       <xi:include href="BusinessAddressBook.xml"/>
       <xi:include href="PersonalAddressBook.xml"/>
</AddressBook>
```

XML ドキュメントがパースされると、上の例で2つある XInclude 要素が、href 属性により指定されたファイルの内容に置換えられます。

xml:base

XMLSpy のXML バリデーターによりXML ドキュメントが読み取られ、XInclude 名前空間内のinclude 要素(これ以降は xi:include と表記)が発見されると、この要素(xi:include)を、xi:include 要素のhref 属性により指定されたXML ドキュメントに置換えます。インクルードされたXML ドキュメントのドキュメント要素(ルート要素)や、XPointer により指定された要素にもxml:base 属性は含まれ、インクルードされたベースURI を保持します。(インクルードされたXML やツリーフラグメントを含む)結果 XML ドキュメントが、スキーマに従う形で妥当でなければならない場合、インクルードされたドキュメントのドキュメント要素(ルート要素)やツリーフラグメントのトップレベル要素を、xml:base 属性を許可するコンテンツモデルにより作成する必要があります。スキーマにより、xml:base 属性がこの要素に対して認められない場合、ドキュメントは妥当でないと評価されます。XMLSpy のスキーマビューにて要素コンテンツモデルのxml:base 属性を定義する方法は、ドキュメンテーションのスキーマビューセクション、xml: プレフィックス属性 220 にて記述されます。

XPointers

XMLSpy ではXInclude 内におけるXPointer がサポートされます。関連するW 3C 勧告は、XPointer Framework ならびに XPointer element() スキーマの2つです。XInclude 要素内のXPointer により、XML ドキュメント全体の代わりに、XML ドキュメント内部でインクルードする箇所を特定することができます。XPointer は、以下にあるように、XInclude 要素内部で使用されます::

```
       <xi:include href="PersonalAddressBook.xml" xpointer="element(usa)"/>
       <xi:include href="BusinessAddressBook.xml" xpointer="element(/1/1)"/>
       <xi:include href="BobsAddressBook.xml" xpointer="element(usa/3/1)"/>
       <xi:include href="PatsAddressBook.xml" xpointer="element(usa)element(/1/1)"/>
```

XPointer のelement() スキーマでは、NCName や子シーケンスを使用することができます。

- 上の例にある最初のxi:include 要素のxpointer 属性ではusa というNCName を伴った要素スキーマが使用されています。XPointer Framework に従うと、このNCName によりusa というID を持つ要素が特定されます。
- 2つ目のxi:include では、element(/1/1) という値がxpointer 属性で使用されています。最初のステップでは、ドキュメントにある最初の子要素(つまりドキュメントが整形式であれば、ドキュメント要素)が特定されます。次のステップでは、最初のステップで特定された要素から見た最初の子要素が特定され、結果としてドキュメント要素にある最初の子要素がインクルードされます。
- 3番目のxi:include にあるxpointer 属性では、NCName と子シーケンスの両方が使用されています。XPointer により、usa という ID を持つ要素の3番目の子要素、その子要素から見た最初の子要素がインクルードされます。.
- 最初に使用したXPointer が期待通りに動作するか分からない場合、4番目の例として示されているxpointer="element(usa) element(/1/1)" という書式を使用することもできます。この方法では、たとえusa というID を持つ要素が存在しない場合でも、ドキュメント要素の最初の子要素が選択されます。これ以外にもバックアップとなる式を追加することができます。各 XPointer は、空白スペース以外で分けることはできず、以下のような書式に従う必要があります: xpointer="element(usa)element(addresses/1)

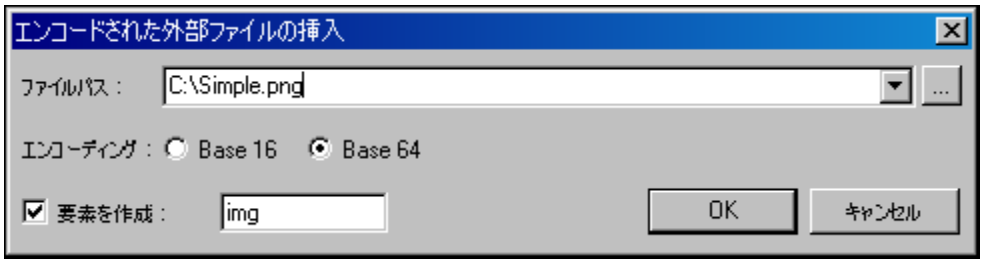
element(/1/1)"。

メモ: element() スキーマはqualified name でサポートされておらず、名前空間バインディングコンテキストは、element() スキーマにて使用されません。

エンコードされた外部ファイルの挿入

エンコードされた外部ファイルコマンドは、テキストビューならびにグリッドビューで使用することができます。このコマンドにより、Base-16 またはBase-64 テキスト形式でエンコードされた外部ファイルを、XML ドキュメント内部の任意の箇所に追加することができます。この機能により外部ファイルを XML ドキュメント内に埋め込むことができます。

「**挿入 | エンコードされた外部ファイル**」コマンドを選択すると、エンコードされた外部ファイルの挿入を行うためのダイアログが表示されます（以下のスクリーンショットを参照）。



埋込みを行う外部ファイルを参照または直接入力します。Base-16 またはBase-64 のエンコーディングを選択してください。エンコードされたテキストを、要素の中に含めて表示する場合、要素を作成チェックボックスにチェックを入れて、作成する要素名をテキストボックスに入力します。要素を作成チェックボックスにチェックが入っていない場合、エンコードされたテキストは、カーソルがある位置に直接挿入されます。

「OK」をクリックすると、選択したファイルをエンコードしたテキストが、（もしそう指定されていれば）要素に含まれる形でカーソル位置に挿入されます。

```
<img ext="png" encoding="xs:base64Binary">
  iVBORw0KGgoAAAANSUhEUgAAABAAAAAQAQMAAAAlPW0iAAAABlBMVEUAAAD/
  //+l2Z/dAAAAM0lEQVR4nGP4/5/h/1+G/58ZDrAz3D/McH8yw83NDDeNGe4U
  g9C9zwz3gVLMDA/A6P9/AFGGFyjOXZtQAAAAAElFTkSuQmCC
</img>
```

上の表示は、PNG イメージファイルをテキストにエンコードしたものです。img 要素がエンコードされたテキストの回りに作成されます。

24.2.8 整形

アイコン

アイコン:	

説明

整形コマンドにより、テキストビューに表示されている XML ドキュメントを再フォーマットすることができます。「**ツール | オプション**」にて表示される*オプションダイアログのビュータブ* 1025 で、*インデントを使用* にチェックが入っているかどうかにより、2つの書式オプションを利用することができます:

- *インデントを使用* がチェックされている：構造が表示されるような形式でドキュメントが再フォーマットされ、各階層レベルがインデントされます。テキストビュー設定 ダイアログ[969] (**ビュー | テキストビュー設定**)によりドキュメントの構造を明確に表示することができます。
- *インデントを使用* がチェックされていない：各行が左寄せになるように再フォーマットされます。

整形を使用する方法

インデントを使用して、XML ドキュメントにおける構造を表示するには、以下の操作を行います:

1. オプションダイアログのビュータブ[1025] (**ツール | オプション**) にて、*インデントを使用* チェックボックスにチェックを入れます。
2. テキストビュー設定 ダイアログ[969] (**ビュー | テキストビュー設定**) から、整形におけるインデントにて使用されるタブの大きさを指定します。
3. オプションダイアログのファイルタブ[1015] (**ツール | オプション**) にて、フォーマット（インデント）を行いたくない要素を入力します。
4. **整形** コマンド（このコマンド）をクリックします。

全ての行が左寄せされた状態にドキュメントを再フォーマットするには、*インデントを使用* チェックボックスのチェックを外してください。

以下の点に注意してください

- このコマンドを使用するには、XML ドキュメントが整形式である必要があります。
- 整形により、ドキュメントが保存される際に、空白やタブが追加されます。
- 整形式が有効になっており（「ツール | オプション | ビュー | インデントを使用」）、テキストビューからグリッドビューへの切り替え、またはその逆が行われると、ドキュメントは自動的に整形されます。XML テキストの整形コマンドを選択する必要はありません。
- 空白の削除[772] コマンドを使用することで、整形により作成された全ての空白（改行とインデント）を削除することができます。

24.2.9 空白の削除

アイコン

アイコン:	

説明

空白の削除 コマンドにより、ドキュメント上にある全ての空白が削除されます。この機能によりファイルの大きさを小さくすることができます。整形[771] コマンド[771] コマンドを使用することで、ドキュメントに改行やインデントを追加することができます。

24.2.10 全て選択

全て選択 (**Ctrl+A**) コマンドにより、ドキュメント全体のコンテンツが選択されます。

24.2.11 検索、次を検索

アイコンとショートカット

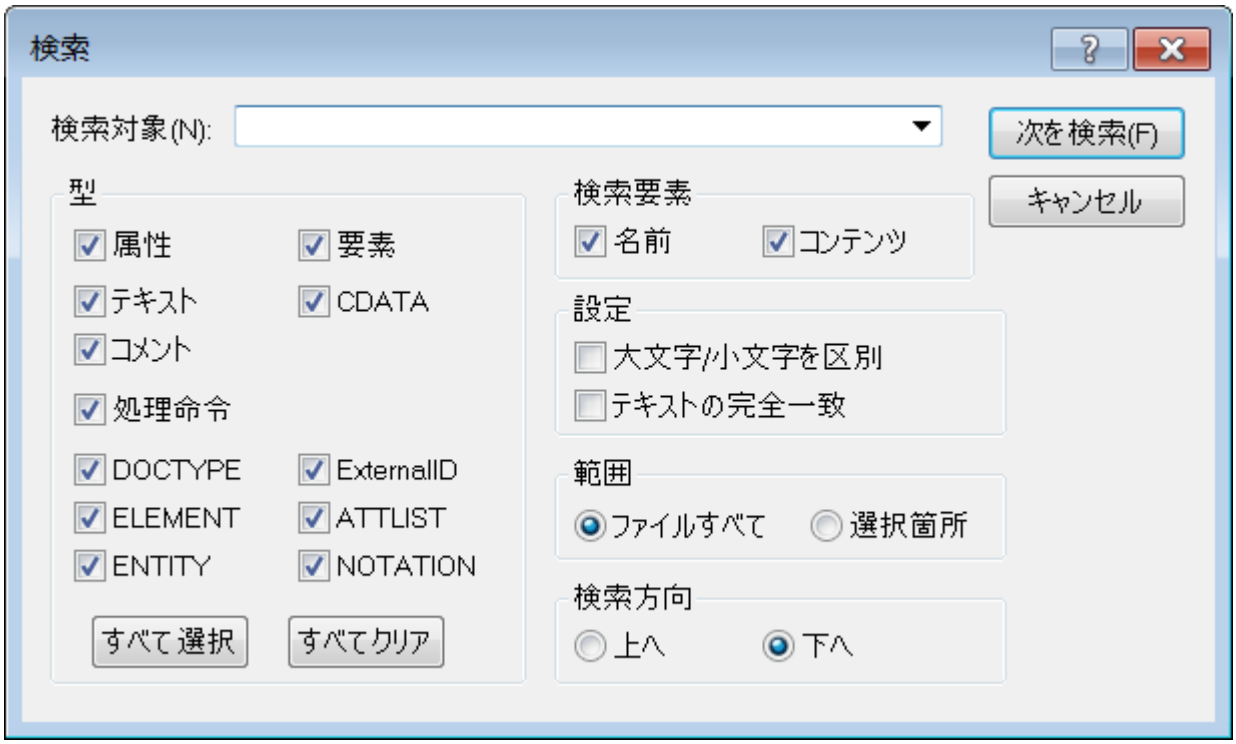
コマンド	アイコン	ショートカット
検索		Ctrl+F
次を検索		F3

検索

検索 コマンド により、検索ダイアログが表示され、検索を行ないたい文字列の入力や、検索に伴う各種オプションを指定することができます。使用しているビューに従って、検索ダイアログには異なるオプションが表示されます。テキストの検索を行うには、検索対象テキストボックスにその文字列を入力するか、コンボボックスを使用して、最後に使用された10個ある検索文字列から選択を行い、検索に必要なオプションを指定します。

検索 と **次を検索** コマンドをプロジェクトウィンドウ内でプロジェクトが選択されるとファイルとフォルダーを検索するために使用することができます。

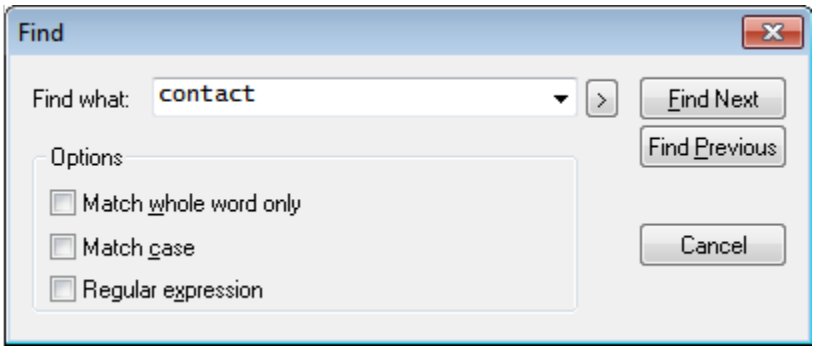
グリッドビュー
グリッドビューでは、次のダイアログボックスが表示されます。必要なオプションを選択、または、ラジオボタンを選択します。使用することのできるオプションについては下で説明されています。



- 型ペインでは、検索対象に含めるXMLドキュメントのノードやコンポーネントを指定することができます。この機能により、特定の型のノードを検索から除外することができます。「全て選択」ボタンにより、ペイン内にある「全てのチェック」ボックスが有効になり、「全てクリア」ボタンにより、全てのチェックボックスが無効になります。

- 検索対象ペインでは、入力された検索文字列に対して、ノード名、ノードのコンテンツ、またはそれら両方を検索対象に含めるか指定することができます。
- 設定ペインでは、検索において大文字と小文字の区別を行うか、または検索対象が入力テキストに完全に一致するかを指定することができます。
- 範囲ペインでは検索の範囲を指定することができます（ファイル全体、または、選択されたテキスト）。
- 検索方向オプションでは、検索が行われる向きを指定することができます。

テキストビュー
編集することのできるテキストビューの検索機能に関しては、以下のセクションを参照してください：テキストビュー 132。（XPath/XQuery ウィンドウ 108 などの）編集することのできないテキストビューの検索機能については下で説明されています。

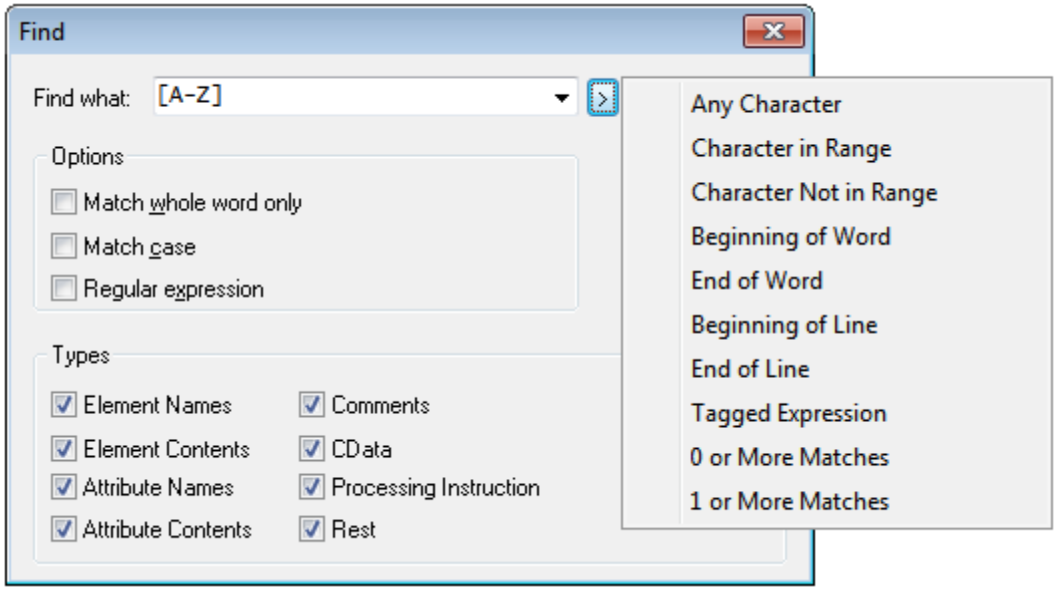


- 単語の完全マッチ：テキスト内にある正確な単語だけが検索のマッチ対象となります。例えば、単語の完全マッチオプションを有効にして、fit という単語で検索を行うと、fit という文字列だけが検索結果として表示され、fitness という単語はマッチしません。
- 大文字・小文字を区別：大文字と小文字を区別した検索を行います（Address とaddress は別物として扱われます）。
- 正規表現：テキストボックスに入力された文字列を正規表現として、検索を行います。正規表現については、正規表現 774 を参照ください。

以下の点に注意してください：

- 検索ダイアログはモードレスになっており、テキストビューを使用中も、ダイアログを開いたままにしておくことができます。ダイアログボックスが開いている間、Enter を押すと、ダイアログボックスが閉じられます。ダイアログボックスを開く前にテキストがマークされると、マークされたテキストは自動的にテキストの検索ボックスに挿入されます。
- 検索ダイアログが閉じられた後、F3 を押下することで現在の検索を継続して行うことができます。Shift+F3 により、反対方向へ検索が行われます。
- 検索対象コンボボックスの右側にある「展開」ボタンをクリックすると正規表現 774 を使用するためのウィンドウが表示されます。

正規表現
正規表現を使用することで、検索の条件をさらに絞り込むことができます。表示されるポップアップリストから正規表現を構築することができます。リストへのアクセスを行うには、検索文字列を入力するフィールドの右にある > ボタンをクリックします。



表示されているアイテムをクリックすることで、それに対応した正規表現が入力フィールドに入力されます。正規表現の構文文字とその説明を以下に示します。

.	全ての文字列にマッチします。これは1つの文字に対応したプレースホルダーです。
\(	マッチのタグ付けを行うための開始マーク。.
\)	タグ付けの終了マーク
\n	ここでのn は1 から9 までの数値を表わしており、上でタグ付けされた範囲を置換えにて使用することができます。例えば、検索文字列がFred¥([1-9])XXX で、置換え文字列がSam¥2YYY である場合、Fred2XXX という文字列がSam2YYY という文字列に置換えられます。
\<	単語の開始位置にマッチします。
\>	単語の終了位置にマッチします。
\x	通常正規表現にて特別な意味を持つx という文字を、通常の文字として扱います。例えば、¥[という文字列は、文字セットの開始ではなく[として認識されます。
[...]	文字のセットを表します。例えば、[abc] という表現により、a、b、c 全ての文字がマッチするようになります。それ以外にも[a-z] という形で範囲を指定することにより、この例では小文字全てをマッチの対象とすることができます。
[^...]	指定された文字セット以外のものにマッチします。例えば、[^A-Za-z] という表現では、アルファベット以外の文字列がマッチの対象になります。
^	行頭を表します(上で述べられている文字のセットで使用される場合を除く)。
$	行末を表します。例えばA+$ という表現で、行末に1つ以上連続して並んでいるA を検索します。
*	0 回以上の繰り返しにマッチします。例えば、Sa*m という表現は、Sm、Sam、Saam、Saaam といった文字列にマッチします。
+	例えば、Sa+m という表現は、Sam、Saam、Saaam といった文字列にマッチします。

メモ 置換 フィールド 776 にて正規表現はサポートされていません。

次を検索

次を検索 コマンドは、最後に実行された検索コマンドを再度実行し、目的のテキスト内で次に出現する検索結果を表示します。

24.2.12 置換

アイコンとショートカット

コマンド	*アイコン*	*ショートカット*
置換		**Ctrl+H**

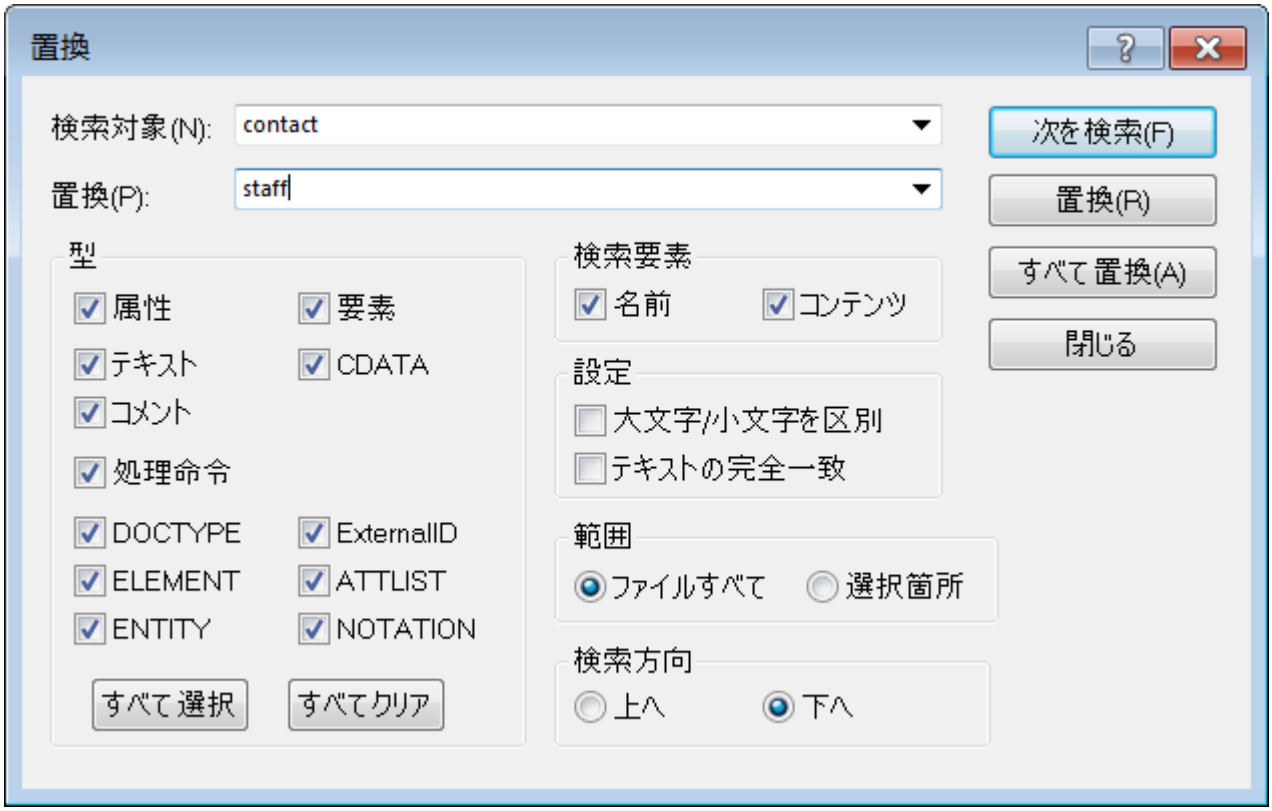
説明

置換コマンドでは、テキスト内のある文字列を別の文字列に置き換えることができます。検索 773 コマンドにて使用されるオプションをここでも使用することができます。使用しているビューの種類により、置換ダイアログに表示されるオプションも異なるものになります。各アイテムを個々に置き換えるか、**全て置換**ボタンを使用することで、検索と置換処理をグローバルに行うことができます。

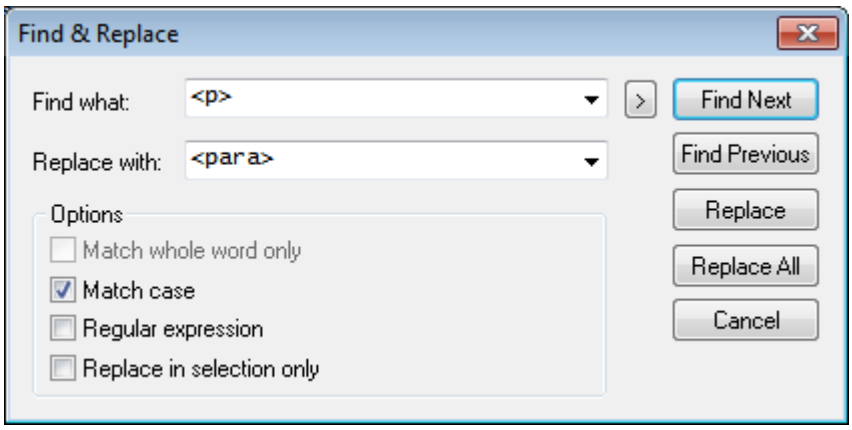
グリッドビュー

グリッドビューでは、**置換** コマンドを選択すると、置換えダイアログが開かれます（*下のスクリーンショット*）。スクリーンショットは、検索 773 コマンドで説明されている異なる検索オプションを表示しています。

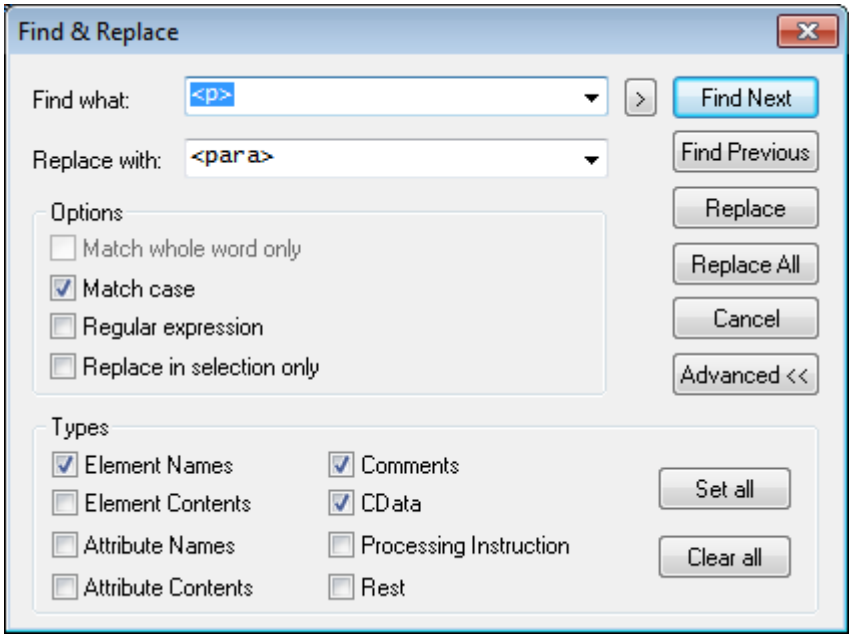
(C) 2015-2021 Altova GmbH



テキストビュー
編集することのできるテキストビューの置換の機能に関しては、以下のセクションを参照してください：テキストビュー[132]。（XPath/XQueryウィンドウ[108]などの）編集することのできないテキストビューの置換機能は下で説明されています。テキストビューでは、置換の選択は、を開きます。コマンド検索と置換ダイアログ（*下のスクリーンショット*）。検索オプションは、検索[773]コマンドの検索のオプションと同じです。選択された範囲内での置換オプションは選択範囲内でのみ検索と置換をおこないます。



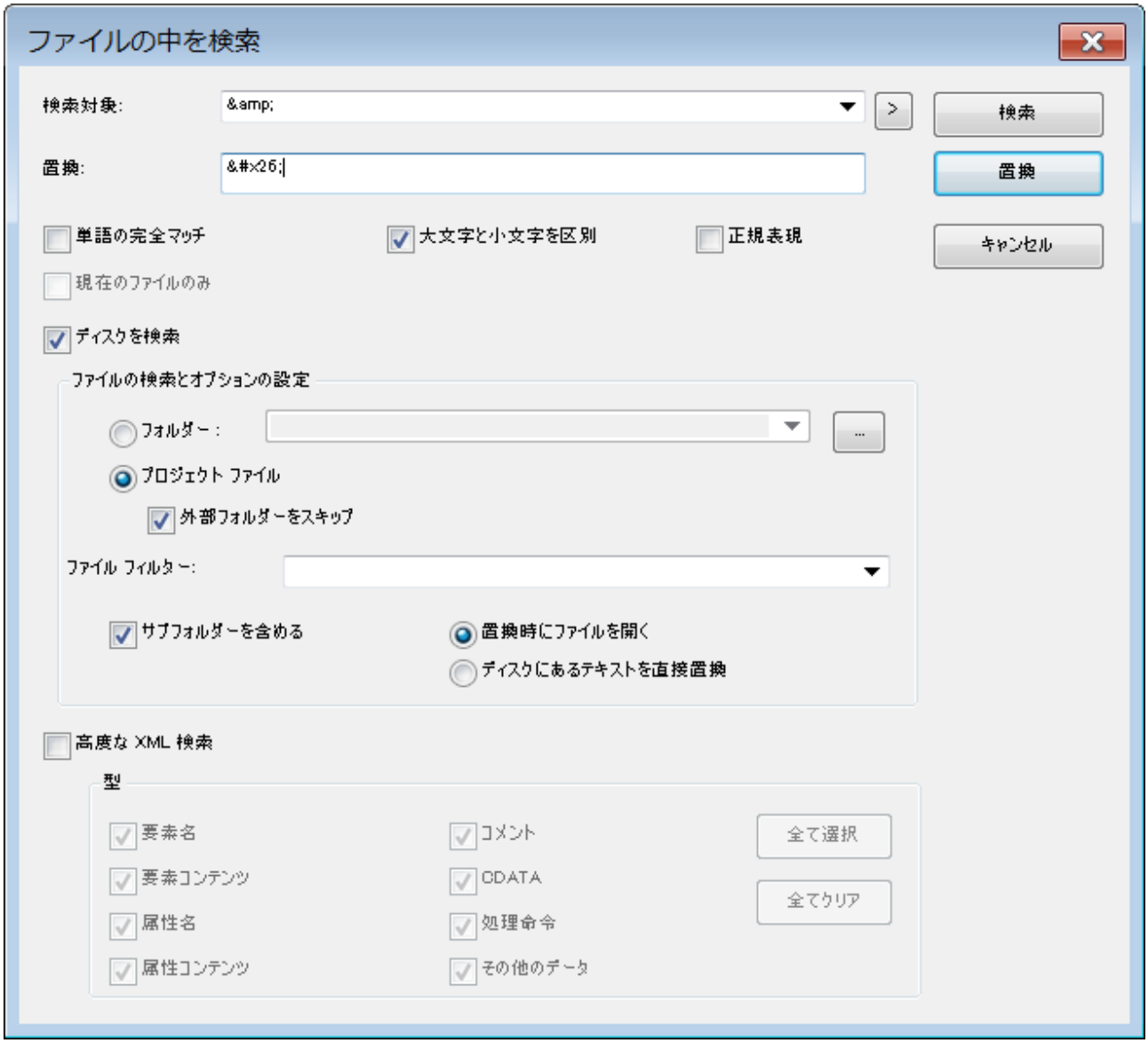
詳細ボタンをクリックすることで、ノードの型の検索のスコープを選択することのできる、型のオプションを指定することができます（次を参照してください：検索[773]コマンド）。



メモ　全て置換コマンドを使用すると、各置換は、単一のオペレーションとして認識されます。従って、全て置換処理の取り消しは、1ステップずつ行うことになります。

24.2.13 ファイル内の検索

ファイルの中を検索コマンドは、数多くのファイルに対して検索と置換を素早く行うことができるパワフルな機能です。コマンドをクリックすることで、ファイルの中を検索ダイアログが表示されます（以下のスクリーンショットを参照）。ファイルの中を検索コマンドは、指定された位置全ての検索ならびに置換処理を一度に行うという点で、検索コマンドとは違っています。検索結果のレポートがファイル内の検索 出力 ウィンドウ 110 に表示されます。しかし検索コマンドの場合は、ユーザーが検索文字列を入力し、検索により見つかったアイテムに対して個々に処理を行うことになります。



検索条件

検索には大きく分けて、(i) 何を検索するのか、そして(ii) どこを検索するのか、という2つの条件があります。検索の対象となるテキストの入力方法については、検索 773 コマンドの記述を参照ください。検索対象テキストボックスにて入力されたテキストが正規表現の場合、正規表現チェックボックスにチェックを入れる必要があります。正規表現に対する入力ヘルパーは、(> ボタン) ボタンからアクセスすることができます。正規表現を使った検索については検索 774 コマンドにある記述を参照ください。

XML ドキュメント内にある検索対象のノードを指定するには、高度な XML 検索チェックボックスにチェックを入れて、目的のノード型のチェックボックスにチェックを入れてください。

現在のファイルのみ チェックボックス、または*ディスクを検索* チェックボックスにチェックを入れることで、どのファイルを検索対象にするか指定することができます。ディスクの検索を選択した場合、検索を行うフォルダーや プロジェクト 782 を(ディスクを検索チェックボックスにチェックを入れた後に)選択することができます。プロジェクトフォルダーが選択されている場合、プロジェクトに追加されている外部フォルダーを検索の対象から

除外することができます。検索の対象となるファイルは、(xml* や xsl* といったかたちで)拡張子とアスタリスクによりフィルタリングすることができます。2つ以上あるファイル拡張子は、コンマやセミコロンで分けることができます(例: xml*;xsl*)。アスタリスクはワイルドカードとして使用することができます。

検索対象に指定された箇所にある検索文字列のインスタンスは、ファイル内の検索 出力バー 110 に表示されます。表示されているアイテムをクリックすることで、そのファイルがテキストビューで表示され、アイテムがハイライトされます。

置換

置換ボタンをクリックすると、全ての検索結果が置換文字列に置き換えられることに注意してください。ファイル内の検索ダイアログにて、*置換時にファイルを開く* が選択されている場合、ファイルがテキストビューで開かれ、*ディスクにあるテキストを直接置換* が選択されている場合、ファイルを開くこと無しに置換が行われます。置換が行われた文字列はファイル内の検索 出力バー 110 に表示されます。表示されているアイテムをクリックすることで、そのファイルがテキストビューで表示され、アイテムがハイライトされます。

メモ: 正規表現は置換フィールドにてサポートされません。

24.2.14 ブックマーク コマンド

アイコンとショートカット

コマンド	*アイコン*	*ショートカット*
ブックマークを挿入/削除する		**Ctrl+F2**
全てのブックマークを削除する		**Ctrl+Shift+F2**
次のブックマークへ移動する		**F2**
前のブックマークへ移動する		**Shift+F2**

ブックマークを挿入/削除する

ブックマークを挿入/削除する コマンドにより、現在のカーソル位置にブックマークを挿入したり、カーソルのある行に既にブックマークが割り当てられている場合にはブックマークを削除することができます。このコマンドはテキストビューでのみ使用することができます。

ブックマークのラインは、以下の方法の内のひとつにより表示されます:

- ブックマーク余白 が有効化されている場合、テキスト左端にあるブックマーク余白に青色のマークが表示されます。
- ブックマーク余白が有効化されていない場合、カーソルが位置している行全体がハイライトされます。

F2 キーにより、ドキュメント内にあるブックマークへ移動することができます。

全てのブックマークを削除する

全てのブックマークを削除する コマンドにより、現在定義されている全てのブックマークを削除することができます。このコマンドはテキストビューでのみ使用することができます。**取り消し**コマンドを使っても、このコマンドを取り消すことができないことに注意してください。

次のブックマークへ移動する

次のブックマークへ移動する コマンドを使用すると、テキストカーソルを次のブックマーク行の行頭へ移動することができます。このコマンドはテキストビューでのみ使用することができます。

前のブックマークへ移動する

前のブックマークへ移動する コマンドを使用すると、テキストカーソルを前のブックマーク行の行頭へ移動することができます。このコマンドはテキストビューでのみ使用することができます。

24.2.15 コメントの IN/OUT

コメントの IN/OUT コマンドはテキストビューで使用することができ、XML テキストフラグメントのコメントアウトやアンコメントを行うことができます。XML ドキュメント内のテキストは、XML のコメント開始ならびにコメント終了タグ（それぞれ <!-- と --> ）によりコメントとして認識されます。XMLSpy では、**コメントの IN/OUT** メニューコマンドを使用することで、これらコメントのタグを簡単に挿入することができます。

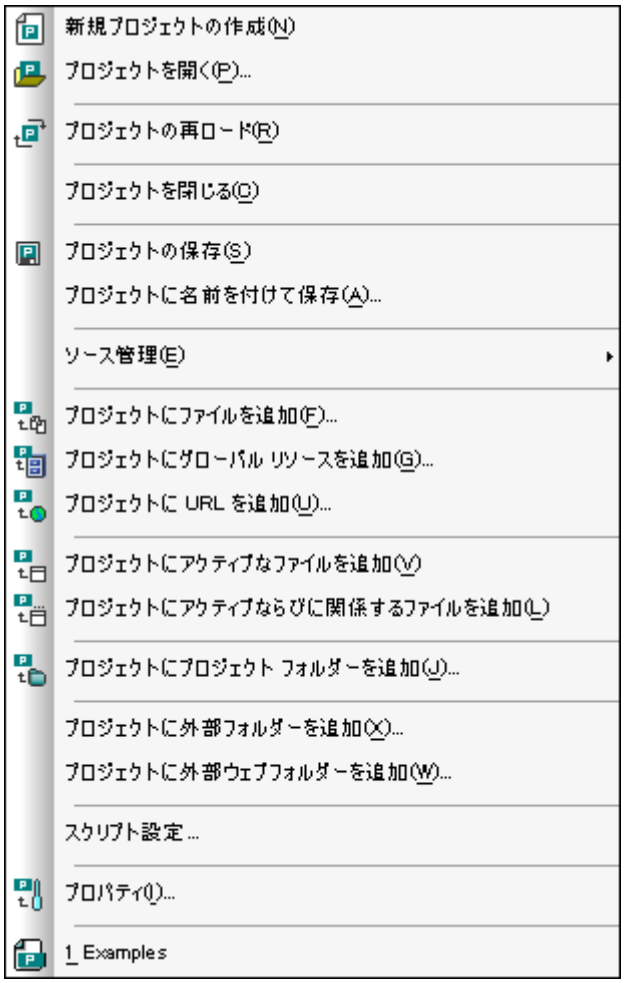
テキストブロックをコメントアウトするには、コメントアウトするテキストを選択し、**編集**メニュー、または選択されたテキストを右クリックすることで表示されるコンテキストメニューから**コメントの IN/OUT** コマンドを選択します。コメントアウトされたテキストはグレーアウトされます（以下のスクリーンショットを参照）。

```
<Department>
    <Name>Administration</Name>
    <Person>
    <Person>
    <Person>
    <!--<Person>
        <First
        <Last></Last>
        <PhoneExt></PhoneExt>
        <EMail></EMail>
        <LeaveTotal></LeaveTotal>
        <LeaveUsed></LeaveUsed>
        <LeaveLeft></LeaveLeft>
    </Person>-->
</Department>
```

コメントされたテキストのブロックをアンコメントするには、**コメントタグを除いて**、コメントアウトされたブロックを選択し、**編集**メニュー、または選択されたテキストを右クリックすることで表示されるコンテキストメニューから、**コメントの IN/OUT** コマンドを選択します。コメントタグが削除され、グレーアウトされていたテキストが通常の色に戻ります。

24.3 プロジェクトメニュー

XMLSpy では、広く使用されているツリービューにより表示される XML プロジェクトを使用することで、複数のファイルや URL を管理することができます。ファイル[799] と URL[799] は拡張子などで フォルダー[800] 内にグループ化することができ、バッチ処理などを簡単に行うことができるようになります。



メモ: プロジェクトに関連したコマンドは、プロジェクト ウィンドウにあるアイテムを右クリックすることで表示されるコンテキスト メニューからアクセスすることができます。

絶対パスと相対パス

各プロジェクトは、.spp という拡張子のついたプロジェクト ファイルとして保存されます。これらのファイルの内容は XML ドキュメント形式で保存されており、通常の XML ファイルとしても編集することができます。プロジェクト ファイルでは、同じ位置より高い場所に対しては絶対パスが、現在のフォルダーやサブフォルダーに対しては相対パスが使用されます。例えば、ディレクトリ構造が以下の様な状況で、

```
|-Folder1
|      |
|      |-Folder2
|             |
|             |-Folder3
```

```
|                    |
|                    |-Folder4
```

.spp ファイルがFolder3 内部にある場合、Folder1 ならびにFolder2 内部のファイルに対する参照は、以下のように記述されます:

c:¥Folder1¥NameOfFile.ext
c:¥Folder1¥Folder2¥NameOfFile.ext

Folder3 ならびにFolder4 への参照は、以下のように記述されます:

.¥NameOfFile.ext
.¥Folder4¥NameOfFile.ext

相対パスの使用を確実なものにするには、作業しているディスク(ドライブ)のルートディレクトリに.spp ファイルを配置してください。

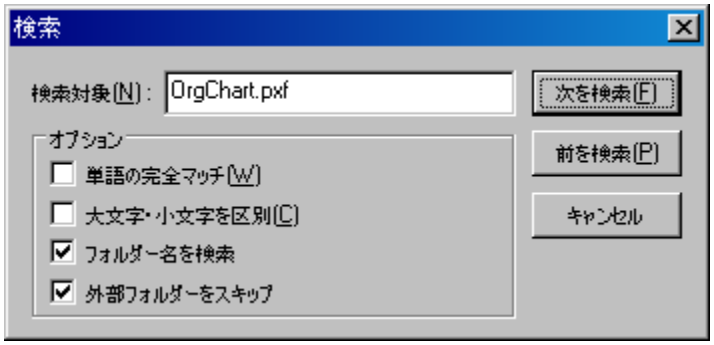
ドラッグアンドドロップ

プロジェクトウィンドウでは、フォルダーをドラッグすることで、別のフォルダーや、同じフォルダー内における別の位置へ、そのフォルダーを移動することができます。ファイルをドラッグすることで別のフォルダーへの移動を行うことはできますが、同一フォルダー内における移動はできません(フォルダー内部のファイルはアルファベット順に表示されます)。更に、Windows ファイルエクスプローラーからプロジェクトウィンドウへ、ファイルのドラッグを行うこともできます。

プロジェクト内の検索

ファイルの名前、またはファイル名の一部からプロジェクト内部にあるファイルやフォルダーを検索することができます。検索が行われると、該当するファイルやフォルダーが1つずつハイライトされた状態で表示されます。

検索を行うには、検索を行うプロジェクトフォルダーをプロジェクトウィンドウにて選択し、「**編集 | 検索**」メニューコマンドを選択するか、Ctrl+F キーを押下します。検索ダイアログが表示されるので、検索を行う文字列を入力し、目的に従うかたちで検索オプションを指定します(以下のスクリーンショットを参照)。検索オプションについては以下を参照ください。



以下の検索オプションが利用できます:

- 単語の完全マッチ: 入力された検索語とファイルまたはフォルダー名が完全に一致するかチェックされ、厳格な検索を行うことができます。ドット(".")の前後に入力された文字列も単語として認識され、例えばファイルの拡張子だけを入力して検索を行うことができます。
- 大文字小文字を区別: 検索語として入力された文字列の大文字と小文字を区別するかどうか指定することができます。
- フォルダー名を検索: フォルダー名を検索の対象に含めます。選択されていない場合、ファイル名だけが検索されます。
- 外部フォルダー[800] を検索の対象に加える、または対象から外す事ができます。外部フォルダーはシステムまたはネットワーク上に実在しているフォルダーで、プロジェクト内部で論理的に作成されたプロジェクトフォルダーとは異なるものです。

検索が正常に行われた場合、最初にマッチしたアイテムがプロジェクトウィンドウにてハイライトされます。検索ダイアログにある**次を検索**ならびに**前を検索**ボタンをクリックすることで、その他のマッチを参照することができます。

プロジェクトの更新

外部 フォルダーに変更が加えられた場合、この変更は、プロジェクトが更新されるまで、プロジェクト ウィンドウ内で反映されません。

コンテキスト メニュー内のグローバル リソース

プロジェクト ウィンドウのフォルダーを右クリックすると、コンテキスト メニューが表示され、その中にあるグローバル リソースファイルを追加メニュー アイテムから、グローバル リソース 660 を追加することができます。メニューコマンドにより、グローバル リソースファイルを追加ダイアログが表示され、現在アクティブなグローバル リソース XML ファイルにある全てのファイル型ならびにフォルダー型のグローバル リソースが表示されます。目的のグローバル リソースを選択すると、選択されたプロジェクト フォルダーに追加されます。

プロジェクトならびにソース管理プロバイダー

XMLSpy プロジェクトをソース管理リポジトリに追加する場合、プロジェクト内にあるファイルを、プロジェクト ファイルが収められている場所以下に配置するようにしてください。

プロジェクト ファイルが位置している場所が、ソース管理リポジトリ内部にあるプロジェクトの**ルート ディレクトリ**になります。これ(プロジェクト ルート ディレクトリ)より上の階層にあるファイルで XMLSpy プロジェクトに加えられたファイルは、(仮にファイルの配置ができたとしても)リポジトリ内部にて予期しない場所に配置される可能性があります。

例えば、上に示したディレクトリ構造で、プロジェクト ファイルが Folder3 に保存され、ソース管理にも配置されると:

- Folder1 に追加されたファイルはソース管理にて配置されない場合があります。
- Folder2 に追加されたファイルは、(プロジェクトフォルダーではなく)リポジトリのルートディレクトリに追加されますが、それでもソース管理以下に配置されます。
- Folder3 ならびに Folder4 に配置されているファイルは、ソース管理以下においても期待されている通りの場所に配置されます。

24.3.1 新規プロジェクト

「**新規プロジェクトの作成**」コマンドを選択すると、XMLSpy にて**新たな**プロジェクトが作成されます。既に他のプロジェクトを開いている場合、そのプロジェクトに属している全てのドキュメントを閉じるかどうか選択することができます。

24.3.2 プロジェクトを開く

「**プロジェクトを開く...**」コマンドでは、XMLSpy における既存のプロジェクトを開きます。既に別のプロジェクトを開いている場合、まずはそのプロジェクトが閉じられます。

(C) 2015-2021 Altova GmbH

24.3.3 プロジェクトの再ロード

プロジェクトの再ロードコマンドは、現在のプロジェクトをディスクから再度ロードします。複数のユーザーが作業をしているような環境では、他のユーザーがプロジェクトに加えた変更を反映するために、プロジェクトの再ロードが必要になる時があります。

メモ: プロジェクトファイル(.spp ファイル)の内容はXMLドキュメントで、その他の一般的な XML ファイルと同様の方法で編集することができます。

24.3.4 プロジェクトを閉じる

プロジェクトを閉じるコマンドにより、現在アクティブなプロジェクトが閉じられます。プロジェクトに変更が加えられている場合、閉じる前にプロジェクトを保存するか選択することができます。プロジェクトに変更が加えられている場合は、プロジェクトウィンドウに表示されているプロジェクト名に、アスタリスクが表示されます。

24.3.5 プロジェクトの保存、名前を付けて保存

プロジェクトの保存コマンドにより、現在のプロジェクトが保存されます。プロジェクトウィンドウをアクティブにして、保存 アイコンをクリックすることでも、プロジェクトの保存を行うことができます。

プロジェクトに名前を付けて保存コマンドは、現在のプロジェクトを新たに指定した別の名前で保存します。

24.3.6 ソース管理

Altova アプリケーション では、Microsoft SourceSafe や、それに準拠したリポジトリがサポートされます。これらシステムのインストール方法については、サポートされているソース管理システム707を参照ください。このセクションでは、XMLSpy 内部からソース管理システムへのアクセスを行う「**プロジェクト | ソース管理**」のサブメニューコマンドについて説明します。

ソース管理機能の概要

アプリケーションプロジェクト内部にあるファイルをソース管理へ配置するメカニズムは以下のようになっています:

1. XMLSpy にて、ソース管理へ配置するファイルが含まれているアプリケーションプロジェクトフォルダーを作成します。通常アプリケーションフォルダーは、プロジェクトファイルが含まれているローカルのフォルダーに対応しています。ローカルフォルダーへのパスは、ローカルパスと呼ばれます。
2. (ソース管理 またはレポジトリとして参照される)ソース管理 システムのデータベース内では、ソース管理下に置かれるファイルを含むフォルダーが作成されます。

3. 「プロジェクト | ソース管理 | ソース管理に追加 791」コマンドにより、アプリケーションプロジェクトファイルがソース管理へ追加されます。
4. チェックインやチェックアウト、ソース管理からファイルを削除といった、ソース管理に関するアクションは、「 プロジェクト | ソース管理 785 サブメニュー 785 コマンドを使用することで行うことができます。このセクションのサブセクションでは、このサブメニューにあるコマンドに関する記述を行います。

メモ: 現在使用しているソース管理プロバイダーを変更するには、(i) ソース管理オプション(ツール | オプション | ソース管理 1040)から行う方法と、(ii) ソース管理の変更ダイアログ(プロジェクト | ソース管理 | ソース管理の変更 798)にて行う方法の2つが用意されています。

メモ: ソース管理プロジェクトは、アプリケーションプロジェクトとは異なるものです。ソース管理プロジェクトはディレクトリ構造に依存しているのに対して、XMLSpy プロジェクトはディレクトリ構造から独立した論理的な構造となっています。

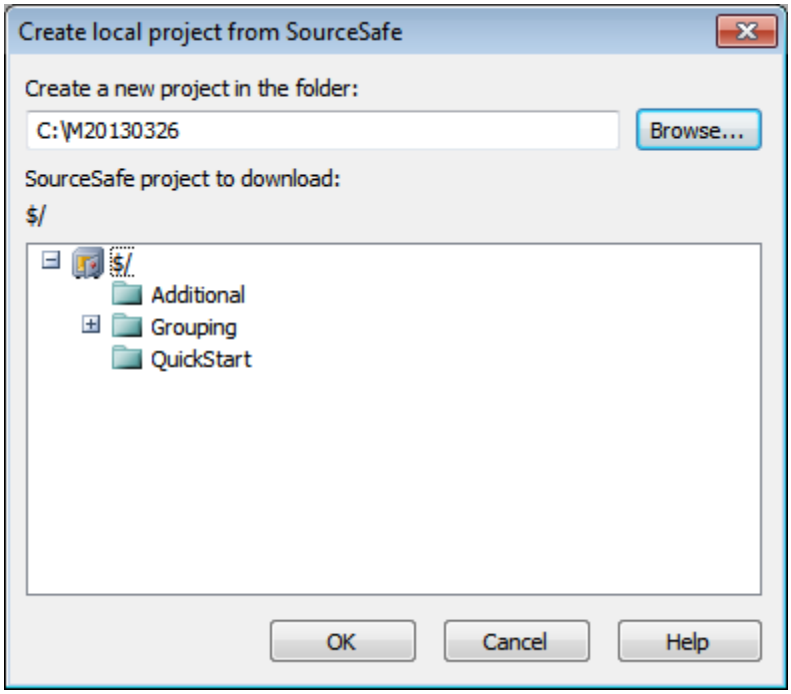
詳細に関しては次のセクションを参照してください: ソース管理 704。

24.3.6.1 ソース管理から開く

ソース管理から開く コマンドにより新規アプリケーションプロジェクトがソース管理下のプロジェクトより作成されます。

新規プロジェクトは以下のとおり作成することができます:

1. 使用されるソース管理システムにより、場合によっては、ソース管理から新規のプロジェクトが作成される前に、プロジェクトからファイルがチェックアウトされないように注意してください。
2. 全てのプロジェクトはアプリケーション内に存在する必要はありません。
3. コマンド「プロジェクト | ソース管理 | ソース管理から開く」を選択します。
4. 現在設定されているソース管理システムは、検証と接続ダイアログをポップアップします。ローカルフォルダーに対応するレポジトリ内のバインドされたフォルダーであるレポジトリに接続を作成します。
5. ポップアップするダイアログ内で、(接続が作成された)レポジトリ内のバインドされたフォルダーのコンテンツがコピーされるローカルフォルダーを参照します(*下のスクリーンショット*)。下のスクリーンショットでは、バインドされたフォルダーは `MyProject` と呼ばれ、`$` サインにより表示されています。ローカルフォルダーは `C:\M20130326` です。



6. 「OK」をクリックします。バインドされたフォルダー(MyProject) のコンテンツは、次のローカルフォルダーにコピーされC:\M20130326.、新規のプロジェクトとして作成される、プロジェクトファイル(.spp ファイル) を選択するようにプロンプトするダイアログがポップアップします。
7. ローカルフォルダーにコピーされる.spp ファイルを選択します。サンプルでは、C:\M20130326 フォルダーにある MyProject.spp です。アプリケーション内にMyProject という名前の新規のプロジェクト が作成され、プロジェクトウィンドウ内に表示されます。プロジェクトのファイルは以下のフォルダー内にあります: C:¥M20130326。

ソース管理のシンボル

フォルダーやファイルにはシンボルが表示されており、それぞれ以下に類似した意味を持ちます。

	チェックイン状態。チェックアウト可能。
	他のユーザーによりチェックアウトされた状態。チェックアウト不可能。
	ローカルにチェックアウト済み。編集してチェックイン可能。

24.3.6.2 ソース管理を有効にする

ソース管理を有効にするコマンドにより、アプリケーションプロジェクトのソース管理を有効化、または無効化することができます。このオプションを選択することで、プロジェクト全体のソース管理を有効化/無効化することができます。ソース管理が有効化されると、さまざまなファイルのチェックインとアウトの状況が抽出され、プロジェクトウィンドウ内に表示されます。

	チェックイン状態。チェックアウト可能。
	他のユーザーによりチェックアウトされた状態。チェックアウト不可能。

	ローカルにチェックアウト済み。編集してチェックイン可能。

24.3.6.3 最新バージョンの取得

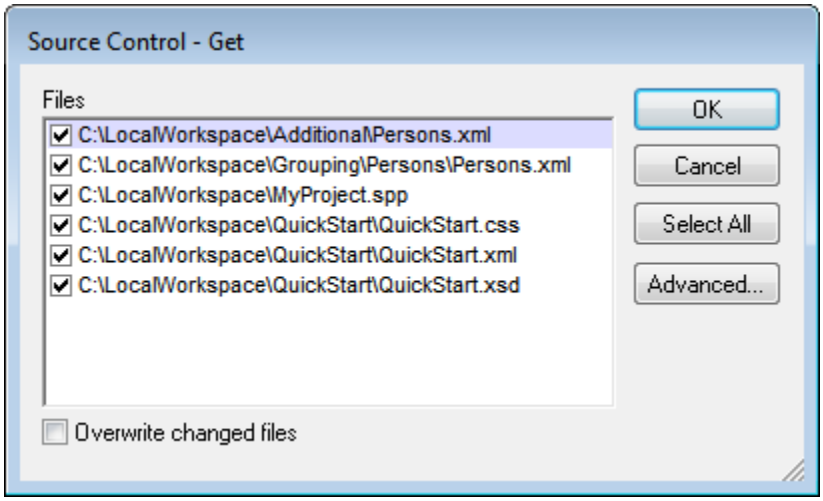
(**プロジェクト | ソース管理** メニュー内の) **最新バージョンの取得**コマンドにより、現在作業中のディレクトリにて選択されたファイルの最新版をソース管理から取得します。ファイルは読み取り専用として取得され、チェックアウトはされません。このコマンドは **Get** 788 コマンド 788 と同じように作動しますが、Get ダイアログは表示しません。
選択されたファイルが現在チェックアウトされている場合、ソース管理システムがどのようにこのような状況を扱うかにより取られるアクションは異なります。通常、ソース管理システムは、チェックアウトされたファイルを置き換える、マージする、またはそのままにするかを問います。

メモ: このコマンドは、フォルダーに対して使用されると、再帰的です。こらは、フォルダー階層構造内の下の全てのファイルに影響を与えます。

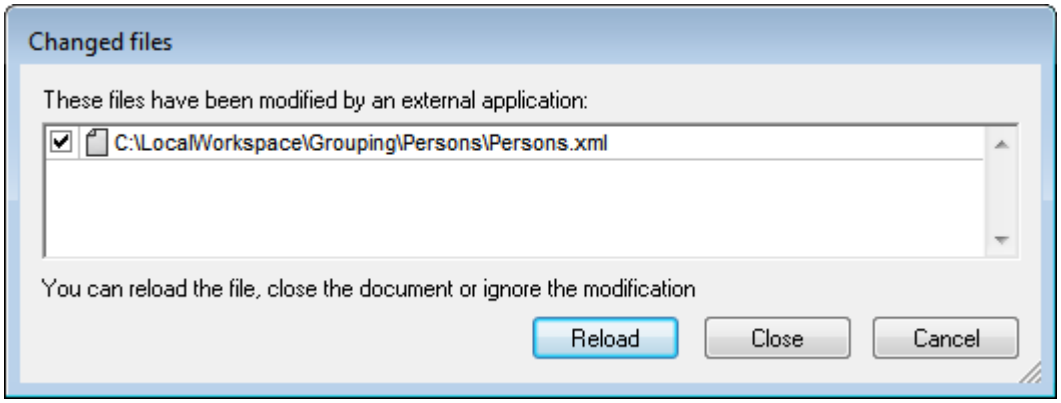
24.3.6.4 フォルダーの取得

(**プロジェクト | ソース管理** メニュー内の) **取得** コマンドはレポジトリからのファイルを読み取り専用 ファイル。として取得します（編集するにはファイルをチェックアウトする必要があります）。取得ダイアログは、「**取得**」コマンドが実行される オブジェクト内のファイル(プロジェクトまたはフォルダー)をリストします（*下のスクリーンショットを参照*）。チェックして取得するファイルを選択することができます。

メモ: ソース管理システムが許可している場合、「**フォルダーの取得**」コマンドによりレポジトリ内の個別のサブフォルダーを選択することができます。

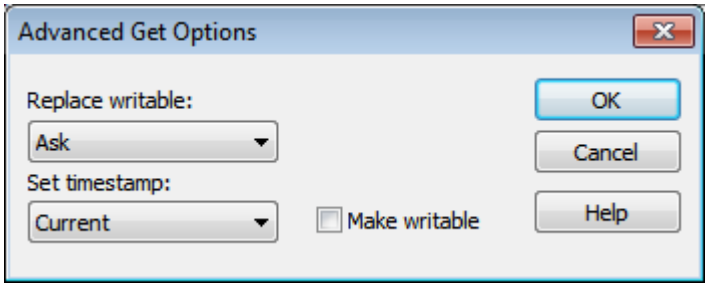


取得ダイアログの下にあるこのオプションをチェックして、変更された、または、チェックアウトされたファイルを上書きすることを選択することができます。「OK」をクリックすると、ファイルが上書きされます。上書きされたファイルが開かれていると、ファイルを再ロード(**再ロード** ボタン)、ファイルを閉める(**閉じる**)、または、ファイルの現在に表示を保持する(**キャンセル**)オプションを問うダイアログがポップアップします（*下のスクリーンショット*）。



高度な取得オプション

高度な取得オプションは取得ダイアログ内の「高度な」ボタンを使用してアクセスすることができますダイアログ（*下のスクリーンショット*）（*このセクションの最初のスクリーンショットを参照*）。



ここでは以下に関する設定をおこなうことができます：（i）チェックアウトされた書き込むことのできるファイルを置き換える。（ii）タイムスタンプ。（iii）取得されたファイルの読み取り専用 プロパティを変更して書き込み可能にする。

24.3.6.5 チェックアウト、チェックイン

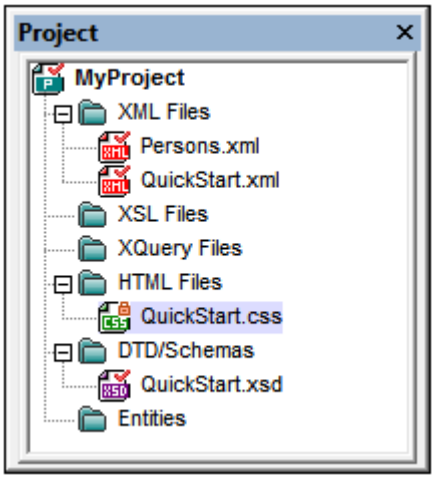
プロジェクト ファイルがソース管理下に置かれると、（プロジェクトウィンドウ内で）ファイルを選択して、「**プロジェクト | ソース管理**」メニュー：「**チェックアウト**」と「**チェックイン**」内の対応するコマンドをチェックして、チェックアウトするか、チェックインするかを選択することができます。

ファイルがチェックアウトされると、ローカルフォルダー内にレポジトリからのコピーが置かれます。チェックアウトされたファイルは編集することができます。ソース管理下にあるファイルがチェックアウトされていない場合、このファイルを編集することはできません。ファイルの編集後、レポジトリに編集を保存することができます。ファイルが保存されない場合でも、レポジトリに変更をチェックインすると変更は保存されます。ファイルがチェックアウトされているかは、アイコン内のチェック、または、ロック済みのシンボルにより表示されています。

ファイルとプロジェクトフォルダーには、下に示されるような特定のシンボルが表示されます。

	チェックイン状態。チェックアウト可能。
	他のユーザーによりチェックアウトされた状態。チェックアウト不可能。
	ローカルにチェックアウト済み。編集してチェックイン可能。

プロジェクト、または、フォルダーをプロジェクト内から選択するには、選択内のファイルを選択します。複数のオブジェクト（ファイルとフォルダー）を選択するには、オブジェクトをクリックして、Ctrl キーを押します。下のスクリーンショットはチェックアウトされたプロジェクトを示しています。ファイル`QuickStart.css` が次にチェックインされています。



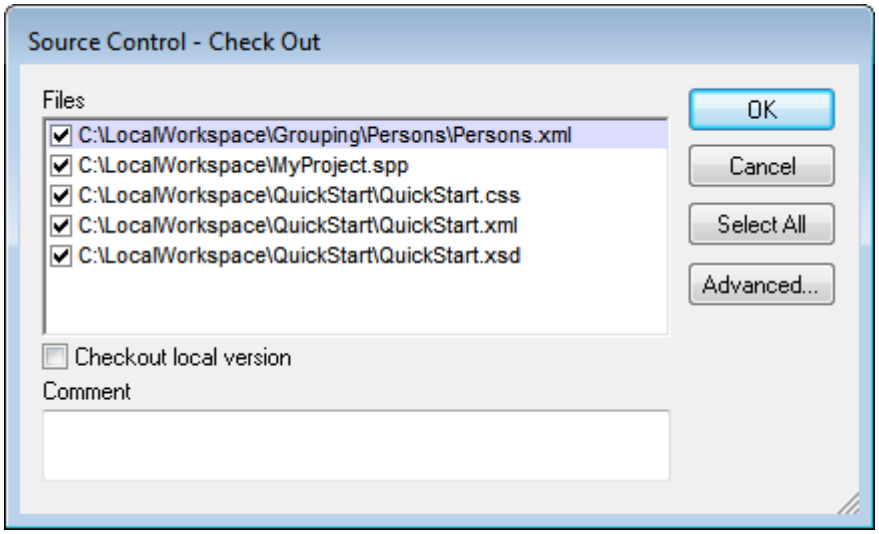
編集の変更を保存、または、拒否する

ファイルをチェックインする場合、ファイルをチェックインしたままにすることができます。ファイルをチェックアウトしたままにして、レポジトリに編集の変更を保存することを意味します。この機能は、定期的にレポジトリに編集の変更を保存して、編集を続ける場合とても役に立ちます。

ファイルをチェックアウトして、編集を行い、ファイルに加えられた変更を拒否する場合、コマンド「**プロジェクト | ソース管理 | チェックアウトの取り消し**」を選択して、レポジトリ内に保存されたドキュメントにバージョンを戻すことができます。

チェックアウト

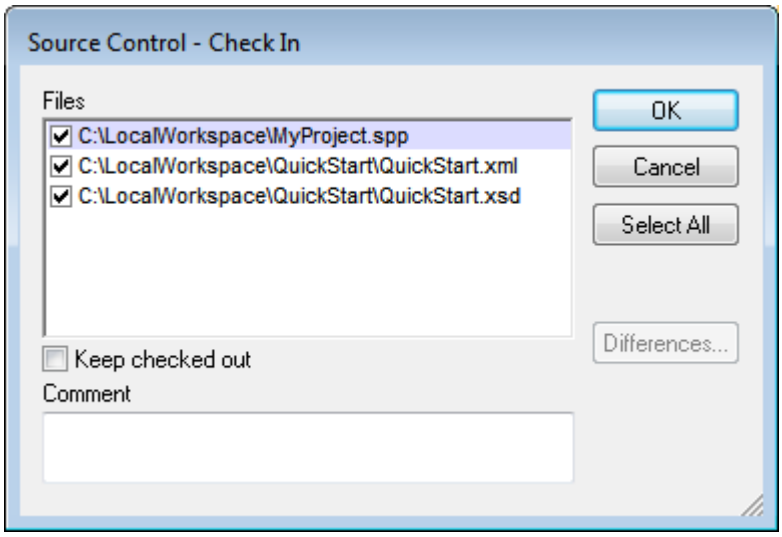
チェックアウト ダイアログ（*下のスクリーンショット*）にて以下を行うことができます: (i) ファイルをチェックアウトする、および、(ii) レポジトリバージョンを選択する、または、ローカルバージョンがチェックアウトされるかを選択する。



(C) 2015-2021 Altova GmbH

チェックイン

チェックインダイアログ（*下のスクリーンショット*）にて以下を行うことができます：（i）ファイルをチェックインする、および、（ii）希望する場合、ファイルをチェックアウトしたままにすることができます。



メモ:　両方のダイアログ（チェックアウトとチェックイン）内で、選択されたオブジェクト（プロジェクトまたはプロジェクトフォルダー）が複数のファイルファイルを含む場合、複数のファイルが表示されます。

24.3.6.6 チェックアウトの取り消し

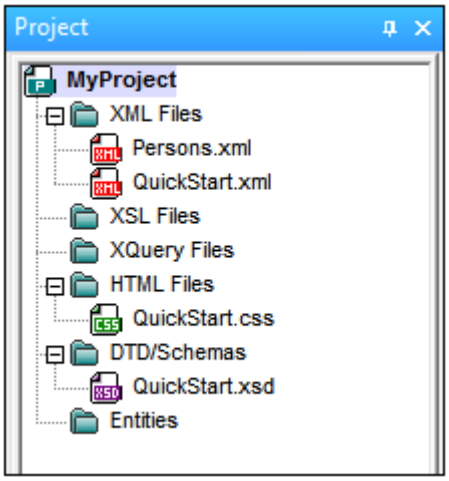
チェックアウトされたファイルが存在し、編集の変更が加えられており、変更を拒否する場合、コマンド「**プロジェクト | ソース管理 | チェックアウトの取り消し**」を使用してレポジトリ内で保存されているドキュメントのバージョンに戻り、ます。

ファイルとプロジェクトフォルダーには、下に示されるような特定のシンボルが表示されます。

	チェックイン状態。チェックアウト可能。
	他のユーザーによりチェックアウトされた状態。チェックアウト不可能。
	ローカルにチェックアウト済み。編集してチェックイン可能。

24.3.6.7 ソース管理へ追加

ソース管理にプロジェクトが追加されると、ファイルを単一に、または、グループとしてソース管理に追加することができます。プロジェクトウィンドウ内でファイルを選択し、**プロジェクト | ソース管理 | ソース管理に追加** をクリックします。複数のファイルを選択するには、追加するファイルをクリック中に**Ctrl** キーを押します。（緑）プロジェクトフォルダー上でコマンドを実行するには、全てのファイルをソース管理のフォルダーとサブフォルダー内に追加します（*下のスクリーンショットを参照*）。



ソース管理にファイルが追加されると、(プロジェクトフォルダー階層構造ではなく)ローカルフォルダー階層構造がレポジトリ内でレプリケートされます。ファイルがサブフォルダー内のローカルフォルダー内で X レベル下にあると、ファイルの親フォルダー他の祖先フォルダーは自動的にレポジトリに追加されます。

ソース管理にプロジェクトから最初のファイルが追加されると、正確なバインドレポジトリ内に作成され、プロジェクトファイル(.spp ファイル)が自動的に追加されます。詳細に関しては、ソース管理へ追加 712 のセクションを参照してください。

ソース管理シンボル

ファイルとプロジェクトフォルダーには、下に示されるような特定のシンボルが表示されます。

	チェックイン状態。チェックアウト可能。
	他のユーザーによりチェックアウトされた状態。チェックアウト不可能。
	ローカルにチェックアウト済み。編集してチェックイン可能。

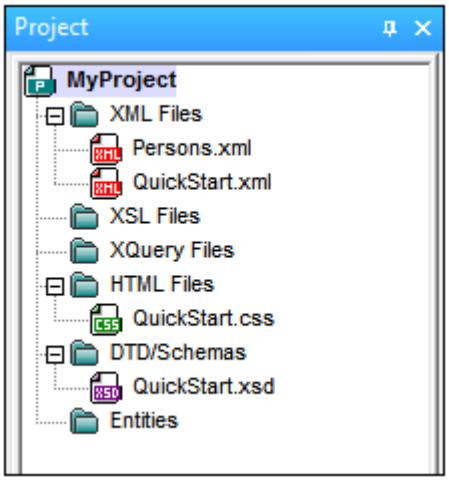
24.3.6.8 ソース管理から削除

ソース管理ファイルからファイルを削除するためには、ファイルを選択して、コマンド「**プロジェクト | ソース管理 | ソース管理から削除する**」をクリックします。以下も削除することができます: (i) フォルダー上の実行コマンドによるプロジェクトフォルダー内のファイルの削除、(ii) **Ctrl** キーを押して複数のファイルを選択しての削除(iii) プロジェクト上の実行コマンドによるプロジェクト全体の削除。

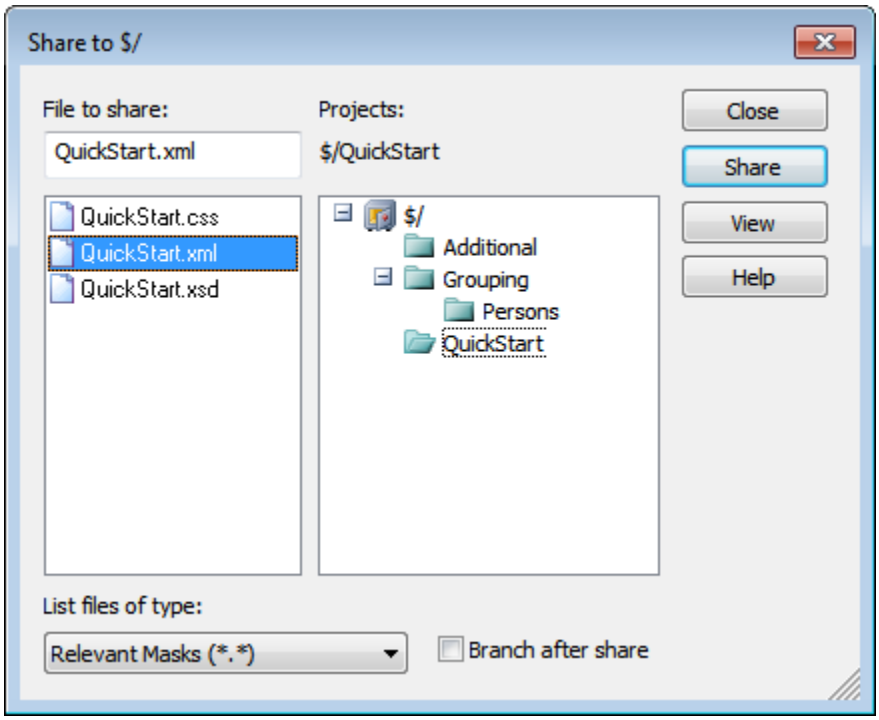
24.3.6.9 ソース管理から共有

ソース管理システムが共有するために使用されている場合、「**ソース管理から共有**」コマンドがサポートされます。複数のローカルの場所でファイルを共有することができます。これらのローカルファイルに加えられた変更は、他の「共有された」バージョンで反映されます。

アプリケーションのプロジェクトウィンドウから最初にプロジェクトを選択します(*下のスクリーンショット内でハイライトされています*)。「**ソース管理から共有**」をクリックします。

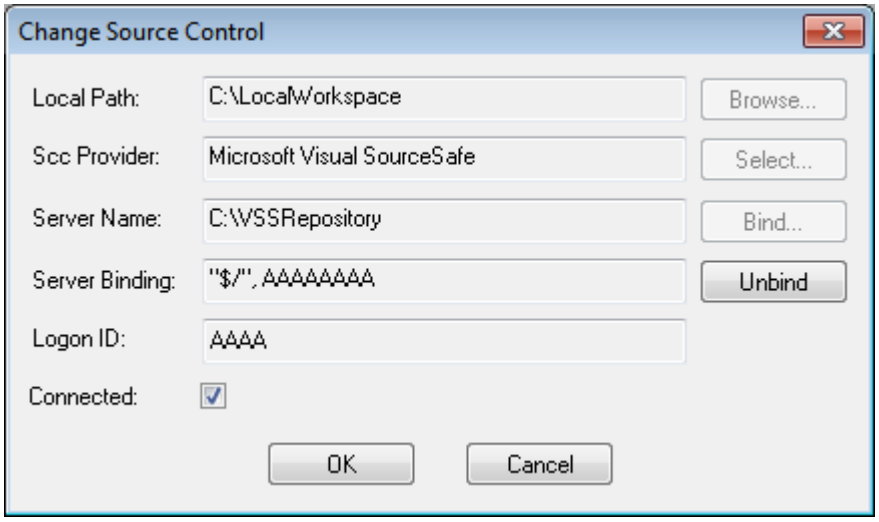


共有するフォルダーダイアログ（*下のスクリーンショット*）がポップアップします。



共有するファイルを選択するには、ファイルが存在するフォルダーが存在する右側のペイン内のプロジェクトツリーを選択します。フォルダー内のファイルは、左側のペインに表示されます。共有するファイルを選択します（複数のファイルは **Ctrl** キーを押して共有するファイルをクリックします）。選択されたファイルは共有するファイルテキストボックス内に（左上）表示されます。「**共有**」をクリックして、選択されたファイルをローカルの共有フォルダーにコピーするために、「**閉じる**」をクリックします。

共有フォルダーは、共有するフォルダーダイアログの名前内に記録されます。上のスクリーンショット内では、（$ サインがローカルフォルダーがバウンドされるレポジトリ内のフォルダーであるため）ローカルフォルダーです。ローカルパスとサーバーのバインドを変更し、ソース管理の変更ダイアログ内で共有フォルダーを確認し設定することができます（*下のスクリーンショットを参照してください*）。

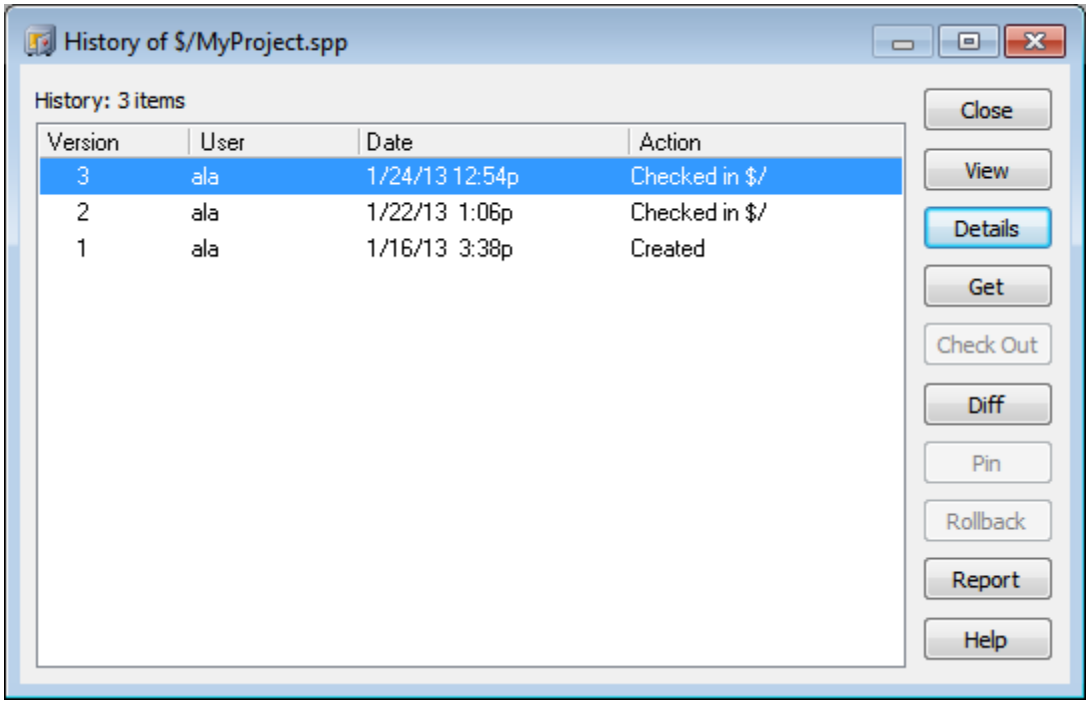


ソース管理システムの詳細に関しては、次を参照してください: ソース管理システムのユーザードキュメント。

24.3.6.10 履歴の表示

履歴の表示コマンドでは、ソース管理以下にあるファイルの履歴を表示して、プロジェクトファイル(`.spp` ファイル) に対して施されたそれまでの変更情報を確認し、差分の表示や、過去バージョンのファイルを取得することができます。以下の操作を行って、ファイルの履歴を確認することができます:

下のスクリーンショットは、Visual SourceSafe ソース管理のシステムの履歴ダイアログを表示しています。`MyProject.spp` ファイルの異なるバージョンを表示しています。



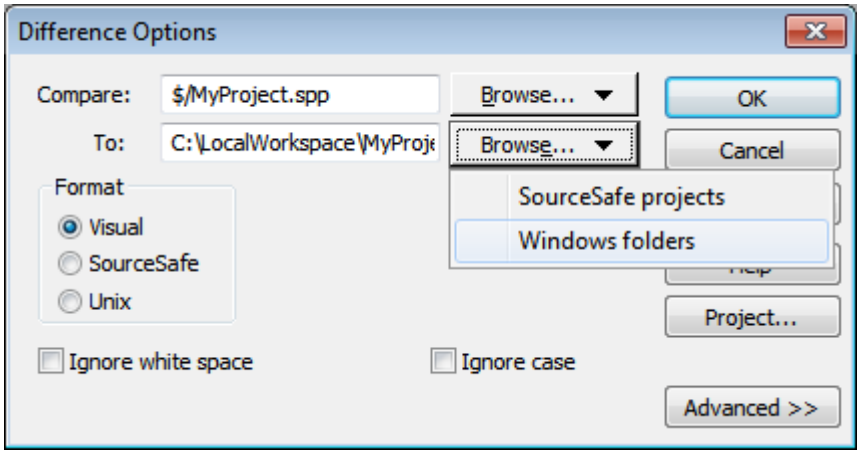
このダイアログボックスでは、目的のファイルを比較したり、特定のバージョンを取得するためのオプションが提供されます。リストにあるエントリーをダブルクリックすることで、そのファイルに関する履歴の詳細情報ダイアログボックスが表示されます。ダイアログ内に表示されるボタンでは、以下の機能が提供されます:

閉じる：このダイアログボックスを閉じます。
表示：ファイルを表示するためのアプリケーションを選択するためのダイアログボックスが表示されます。
詳細情報：現在アクティブになっているファイルのプロパティ797 を確認するためのダイアログボックスが表示されます。
取得：バージョンリストに表示されているファイルのバージョンを取得し、現在作業中のディレクトリに配置します。
チェックアウト：前バージョンのファイルをチェックアウトすることができます。
差分：差分表示オプション795 ダイアログボックスを開き、ファイルバージョンにおける差分を表示する際に使用される表示オプションを設定することができます。
ピン設定：ファイルのバージョンにピンを打つことで、特定のファイルバージョンを差分に使用することができます。
ロールバック：選択されたバージョンのファイルへロールバックします。
レポート：プリンター、ファイル、またはクリップボードへ送ることができる履歴レポートを作成します。
ヘルプ：ソース管理プロバイダープラグインのオンラインヘルプを開きます。

24.3.6.11 差分の表示

「**差異の表示**」コマンドは、プロジェクトウィンドウ内でファイルが選択されると有効化されます。プロジェクトファイル（.spp ファイル）を選択するには、プロジェクトウィンドウ内でプロジェクトのタイトルを選択します。「**差異の表示**」コマンドは、ソース管理システムの差異ツールを開始し、Altova アプリケーションからファイル間の差異が直接チェックできるようになります。

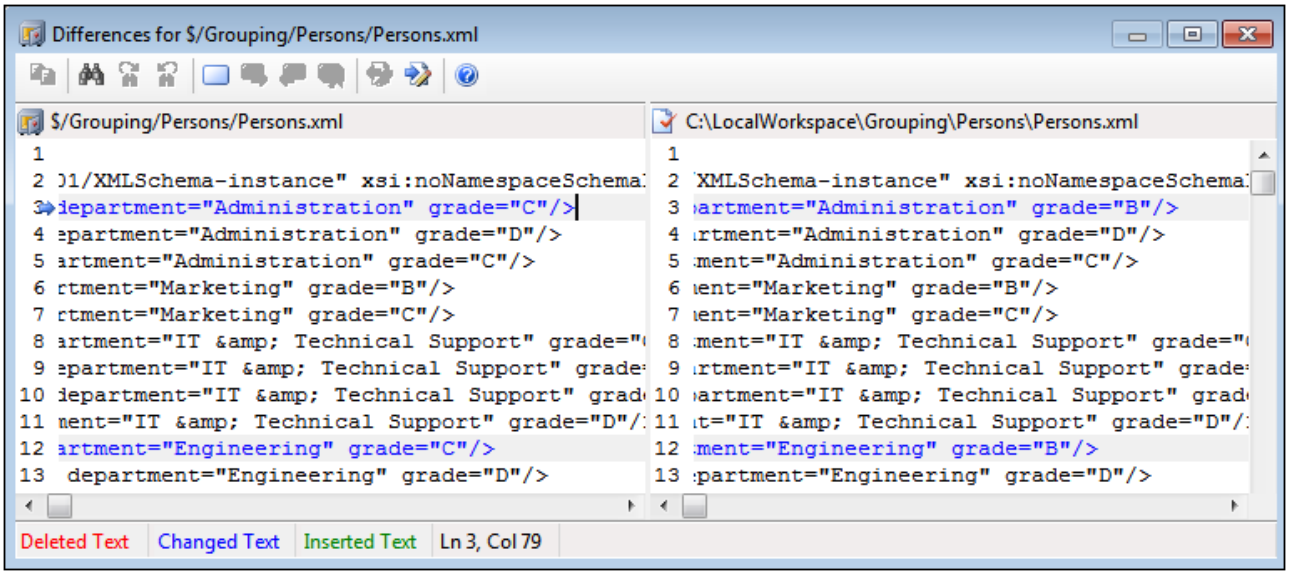
下のスクリーンショットは、Visual SourceSafe ソース管理システムの差異ツールを表示しています。



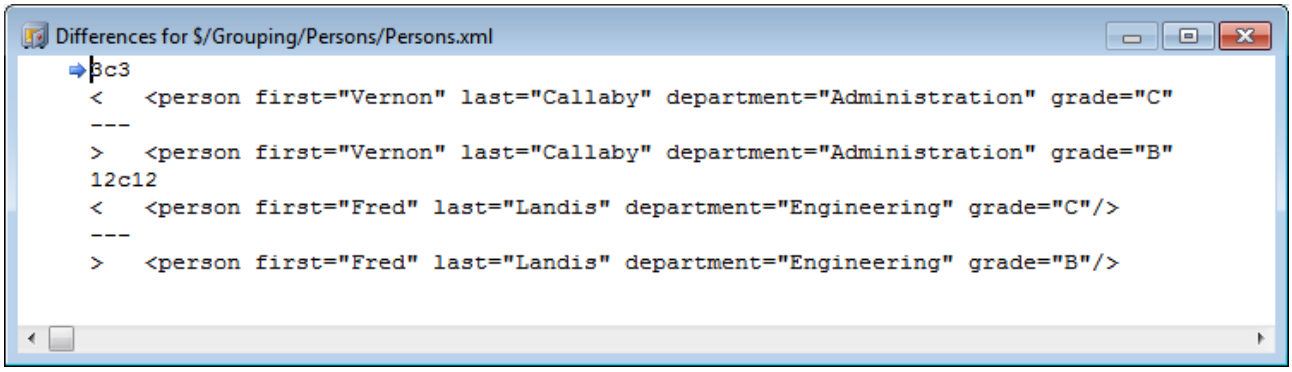
レポジトリとローカル バージョン がそれぞれ比較とテキストフィールドにデフォルトで表示されます。他のファイルの参照の方法は、以下のとおりです:

1. 「参照」ボタンドロップダウンリストから、(レポジトリフィールドを参照するために) ソースセーフ プロジェクト または (ローカルフォルダーを選択するために) W indows フォルダーを選択します。
2. ファイルを参照し、選択します。

適切なエントリーを選択し、「OK」により確定します。差異の結果が個別のウィンドウに表示されます。下のスクリーンショットは、2つのフォーマットのチェックインの結果を表示しています。



上のスクリーンショットは、Visual SourceSafe 差異の結果を Visual フォーマットで表示して折、したのスクリーンショットは、Unix フォーマットでの結果を表示しています (*上のオプション ダイアログを参照してください*)。2つの差異の種類が存在し、それぞれが C から B. へのグレードの変更を表示しています。

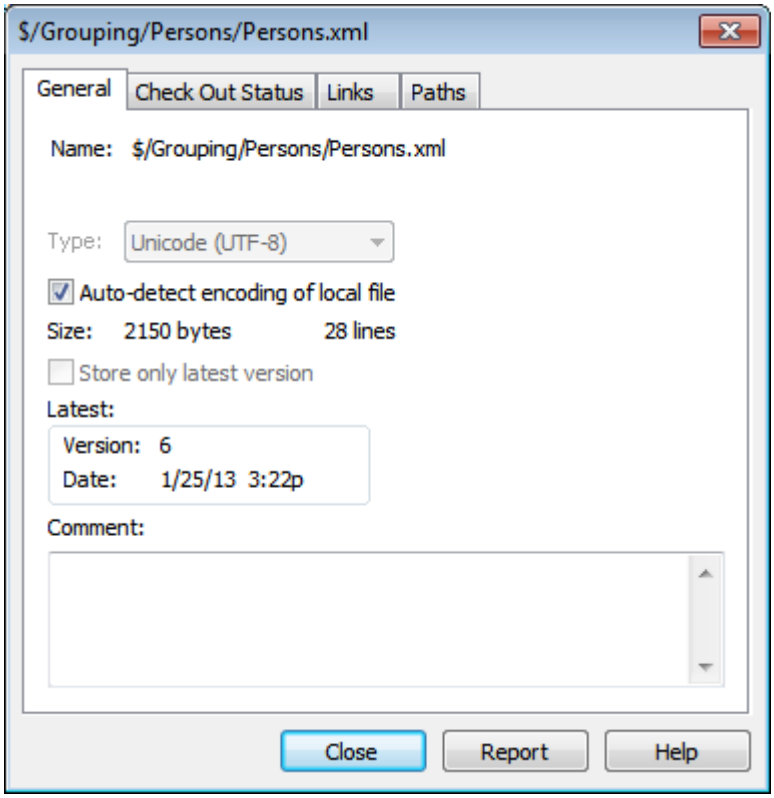


ソース管理システムの差異の処理に関しては、次を参照してください: 製品のユーザードキュメント。

24.3.6.12 プロパティの表示

プロパティの表示コマンドにより、現在選択されているファイルのプロパティが表示されます(以下のスクリーンショットを参照)。表示されるプロパティは、使用しているソース管理プロバイダーにより変化します。下のスクリーンショットは、Visual SourceSafe がアクティブなソース管理システムであることを示しています。

このコマンドは、単一のファイルのために有効化されていることに注意してください。



このコマンドは単一のファイルに対してしか使用できないことに注意してください。

24.3.6.13 ステータスを最新の状態に更新

「**ステータスを最新の状態に更新**」コマンドにより、プロジェクトファイル全てのステータスが、現在のステータスに関係なく最新の状態に更新されます。

24.3.6.14 ソース管理マネジャー

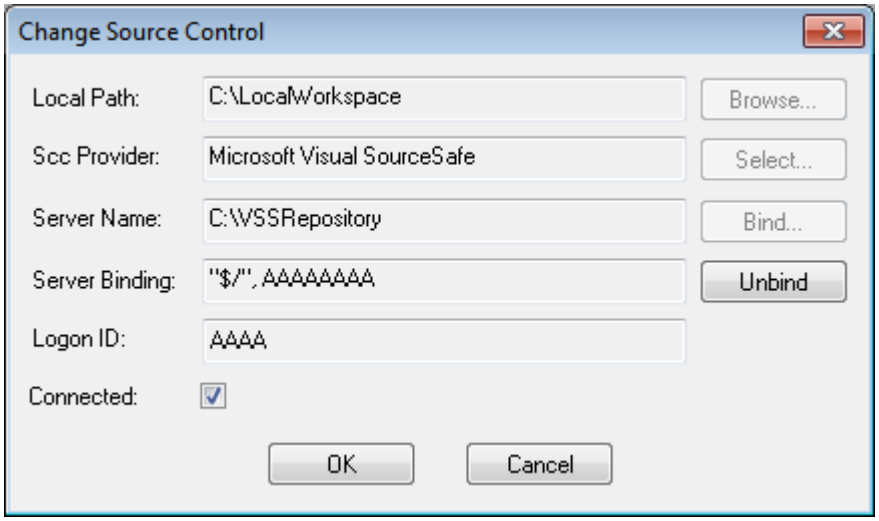
「**ソース管理**」コマンドにより、ソース管理ソフトウェアがネイティブのユーザーインターフェースにて起動されます。

24.3.6.15 ソース管理の変更

アクティブな アプリケーションプロジェクト がソース管理データベースに接続するために現在のバインドが使用されるため、現在のバインドは正確である必要があります。アプリケーションプロジェクトファイル(.spp ファイル)がローカルパスフォルダー内に存在し、レポジトリ上のバインドされたフォルダーがプロジェクトのファイルが保管されているデータベースである必要があることを意味します。通常、バインドされたフォルダーとそのサブ構造は、ローカルワークプレースとそのサブ構造に対応しています。

ソース管理の変更 ダイアログ内でソース内で管理システム(*SCC プロバイダー*)、ローカルフォルダー(*ローカルパス*)、およびレポジトリバインド(*サーバー名* と *サーバーのバインド*).を変更することができます(*下のスクリーンショット*)。

現在のバインドのバインドを解除した後のみに設定を変更することができます。「**バインドの解除**」ボタンを使用して現在のバインドを解除します。この後、全ての設定を編集することができます。



ソース管理の変更設定 は、以下のとおりです:

1. 「**参照**」ボタンを使用してローカルフォルダーを参照し、インストールされたソース管理 システムから「**選択**」ボタンを選択します。
2. レポジトリデータベースにローカルフォルダーをバインドすることができます。「**バインド**」ボタンをクリックしてこれを行います。ソース管理システムの接続 ダイアログが表示されます。
3. *ログイン ID* を入力すると、ソース管理システムにパスされます。それ以外の場合、接続ダイアログ内に詳細を入力します。

(C) 2015-2021 Altova GmbH

4. ローカルフォルダーにバインドするレポジトリ内のデータベースを選択します。この設定 は、1つ以上のダイアログを完成する必要がある可能性があります。
5. 設定が作成された後、「OK」ボタンをクリックしてください。

24.3.7 プロジェクトにファイルを追加

「**プロジェクト | プロジェクトにファイルを追加**」コマンドにより、ファイルを現在のプロジェクトへ追加することができます。このコマンドを使用して、ファイルをプロジェクト内部にある任意のフォルダーに追加することができます。開くダイアログでは、単一のファイルを選択したり、**Ctrl** を押下したままクリックすることで複数のファイルを選択することができます。プロジェクトにファイルを追加している場合は、プロジェクトプロパティ[808] ダイアログボックスで定義されているファイル拡張子をベースに、適切なフォルダーにファイルが配置されます。

24.3.8 プロジェクトにグローバルリソースを追加

「**プロジェクト | プロジェクトにグローバルリソースを追加**」コマンドにより、グローバルリソースを追加ダイアログが表示され、プロジェクトに追加するファイルまたはフォルダーのグローバルリソースを選択することができます。ファイル型のグローバルリソースが選択されると、プロジェクトプロパティ[808] ダイアログボックスで定義されているファイル拡張子をベースに、ファイルが適切なフォルダーに追加されます。フォルダー型のグローバルリソースが選択されると、そのフォルダーがファイルを開くダイアログにて開かれ、ファイルを選択するよう促されます。選択されたファイルは、プロジェクトプロパティ[808] ダイアログボックスで定義されているファイル拡張子をベースに、適切なフォルダーに追加されます。グローバルリソースに関する詳細は、ドキュメンテーションのグローバルリソースのセクションを参照ください。

24.3.9 プロジェクトに URL を追加

「**プロジェクト | プロジェクトにURL を追加**」コマンドでは、現在のプロジェクトにURL を追加することができます。プロジェクトにURL を追加することで、URL のターゲットオブジェクトをプロジェクトに追加することができます。URL やURL オブジェクトが含まれるフォルダーに対してバッチ処理が行われると、XMLSpy によりURL からドキュメントが取得され、指定されたオペレーションが行われます。

24.3.10 プロジェクトにアクティブなファイルを追加

「**プロジェクト | プロジェクトにアクティブなファイルを追加**」コマンドにより、アクティブなファイルを現在のプロジェクトに追加することができます。ハードディスクやURL からファイルを開くと、このコマンドによりそのファイルを現在のプロジェクトに追加することができます。

24.3.11 プロジェクトにアクティブならびに関係するファイルを追加

「**プロジェクト | プロジェクトにアクティブならびに関係するファイルを追加**」コマンドを選択すると、現在アクティブな XML ドキュメントと、それに関係する全てのファイルがプロジェクトに追加されます。DTD やスキーマをベースにした XML ドキュメントを処理している場合、このコマンドを選択することで XML ドキュメントだけでなく、関連する全てのファイル(例えば、DTD ならびに DTD が参照する全ての外部エンティティ)が現在のプロジェクトに追加されます。

メモ: : 処理命令により参照されるファイル(例えば XSLT ファイル)は、関係するファイルとしては認識されません。

24.3.12 プロジェクトにアクティブならびに関係するフォルダーを追加

プロジェクト | プロジェクトにプロジェクトフォルダーを追加する コマンドは、新規のフォルダーを現在のプロジェクトに追加します。このコマンドを使用して、新規のフォルダーをプロジェクトフォルダーの現在のプロジェクトまたはサブフォルダーに追加することができます。プロジェクトウィンドウ内のフォルダーを右クリックして、コンテキストメニューからこのコマンドにアクセスすることができます。

メモ: プロジェクトフォルダーをドラッグして、プロジェクトフォルダー、または、プロジェクト内の他の場所にドロップすることができます。更に、フォルダーを Windows (ファイル) Explorer からドロップして、プロジェクトフォルダーにドロップすることもできます。
メモ: プロジェクトフォルダーは緑で示され、外部フォルダー 800 は黄色で示されています。

24.3.13 プロジェクトに外部フォルダーを追加

「**プロジェクト | プロジェクトに外部フォルダーを追加**」コマンドにより、外部フォルダーを現在のプロジェクトへ追加することができます。このコマンドによりローカルまたはネットワークフォルダーを現在のプロジェクトへ追加することができます。プロジェクトウィンドウにあるフォルダーを右クリックすることで表示されるコンテキストメニューからも、このコマンドへアクセスすることができます。

メモ: 外部フォルダーは黄色で、プロジェクトフォルダー 800 は緑で表示されます。

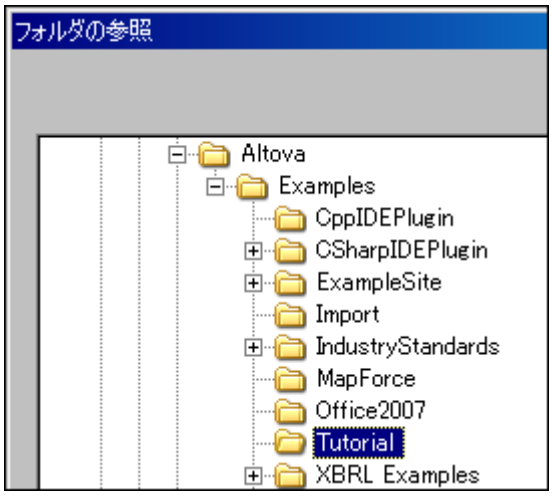
メモ: 外部フォルダー内部に収められているファイルを、ソース管理内に収めることはできません。

外部フォルダーをプロジェクト追加する

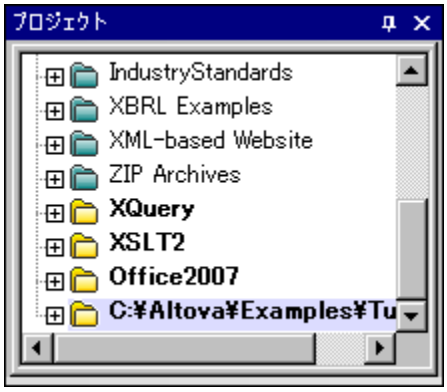
外部フォルダーをプロジェクトへ追加するには、以下の操作を行ってください:

1. メニューオプションから「**プロジェクト | プロジェクトに外部フォルダーを追加**」を選択します。
2. フォルダーの参照ダイアログボックスにて追加するフォルダーを選択し、「**OK**」により確定します。

(C) 2015-2021 Altova GmbH



選択されたフォルダーがプロジェクトウィンドウに表示されます。

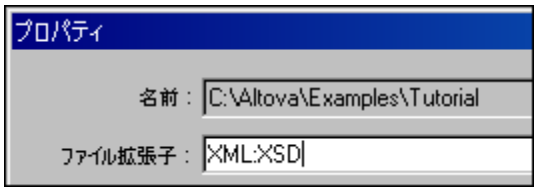


3. プラスアイコン（＋）をクリックして、フォルダーのコンテンツを確認します。

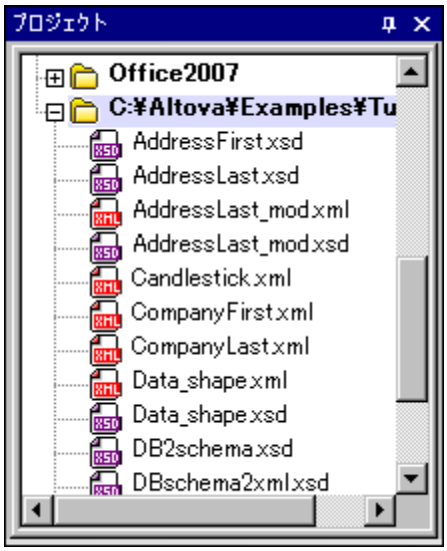
フォルダーのコンテンツをフィルタリング

フォルダー内部にあるコンテンツをフィルタリングするには、以下の操作を行ってください:

1. ローカルフォルダーを右クリックして、表示されるメニューオプションから「**プロパティ**」を選択します。プロパティダイアログボックスが表示されます。



2. ファイル拡張子フィールドをクリックし、表示したいファイル拡張子を入力します。セミコロンにより、各ファイルタイプを分けることができます（例えば、この例ではXMLとスキーマXSD が分けられています）。
3. 「**OK**」をクリックして確認します。



プロジェクトウィンドウにはXML とXSD ファイルだけが表示されるようになります。

外部フォルダーの検証

外部フォルダーの整形式チェックや検証を行うには、以下の操作を行ってください:

1. 外部フォルダーからチェックを行うファイルを選択します。
2. フォルダーを選択して、メニューコマンド[XML | 整形式をチェック] または[XML | XML の検証]をクリックします(それぞれホットキー[F7] または[F8]が対応します)。フォルダーの下の可視化されているファイルがチェックされます。ファイルが形式に誤りがある、または、無効な場合このファイルはメインウィンドウで開かれ編集することが可能になります。
3. エラーを正して再チェックのためにに検証プロセスを実行します。

プロジェクトフォルダーの更新

ローカルやネットワーク上にファイルが追加されたり、削除されることもあります。フォルダーの表示を更新するには、外部フォルダーを右クリックして、ポップアップメニューオプションから「**最新の状態に更新**」を選択します。

外部フォルダーやファイルの削除

外部フォルダーを選択して「**削除**」キーを押下することで、プロジェクトウィンドウからフォルダーを削除することができます。ほかにも、外部フォルダーを右クリックして「**削除**」コマンドを選択することもできます。これらの操作によりプロジェクトウィンドウからフォルダーが削除されますが、ハードディスクやネットワーク上にあるフォルダーが削除されることはありません。

外部フォルダーにあるファイルを削除するにはハードディスクやネットワークから物理的に削除する必要があります。プロジェクトになされた変更を確認するには、外部フォルダーを右クリックして、「**最新の状態に更新**」を選択してください。

メモ: ドラッグアンドドロップにより、外部フォルダーを他のプロジェクトフォルダーや、プロジェクト内の別の場所へ移動することができます(但しほかの外部フォルダーへは不可)。更に、Windows (ファイル)エクスプローラーから、外部フォルダーを(外部フォルダー以外の)プロジェクトフォルダーへドラッグすることでも、フォルダーを追加することができます。

(C) 2015-2021 Altova GmbH

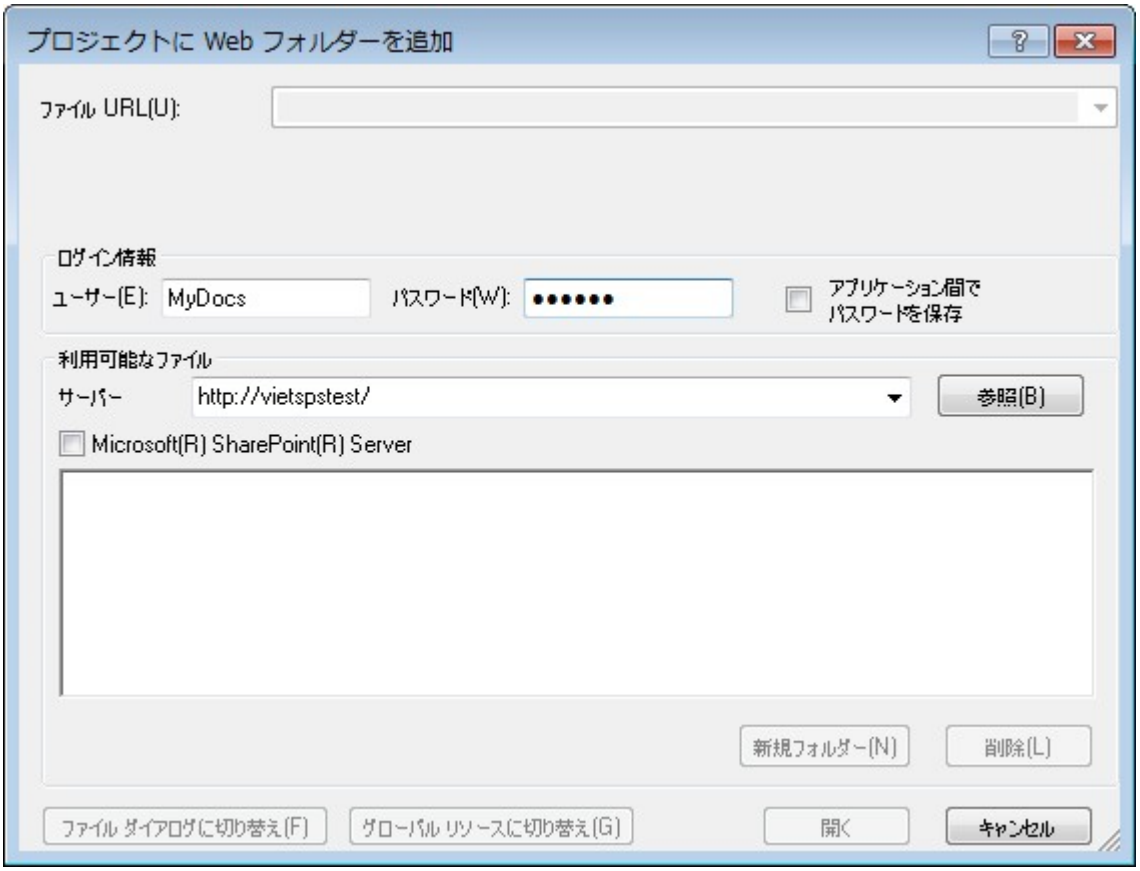
24.3.14　プロジェクトに外部ウェブフォルダーを追加

このコマンドは現在のプロジェクトに新規の外部 W eb フォルダーを追加します。プロジェクトウィンドウ内のフォルダーを右クリックすることによりコンテキストメニューからこのコマンドにアクセスすることができます。外部フォルダーに含まれるファイルは、ソース管理の下に置くことはできません。

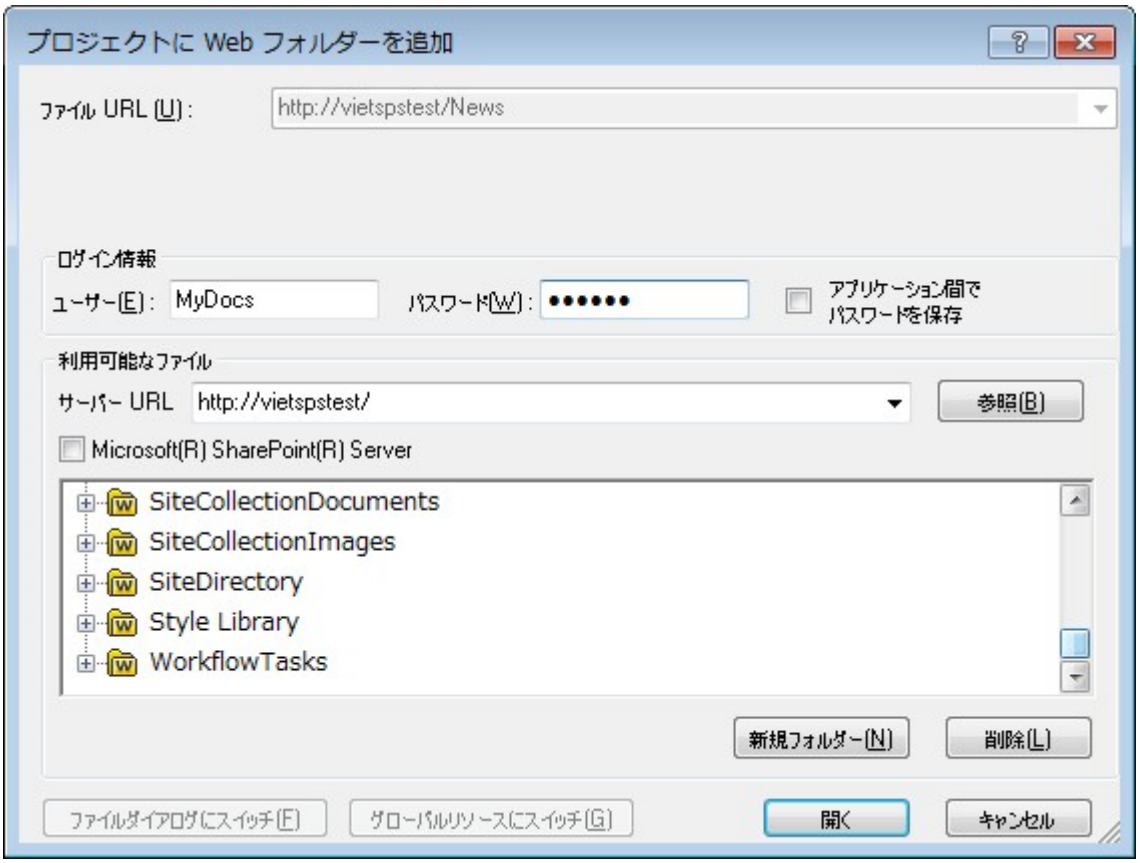
外部 W eb フォルダーをプロジェクトに追加する

外部 W eb フォルダーをプロジェクトに追加するには、以下を行います:

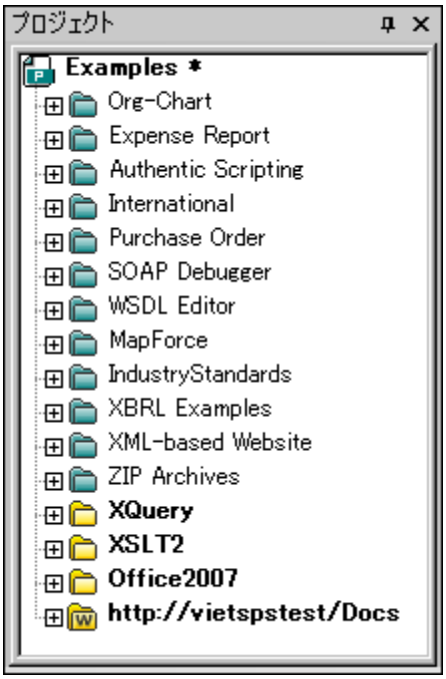
1. メニューオプション「**プロジェクト | 外部 W eb フォルダーをプロジェクトに追加する**」を選択します。プロジェクトにW eb フォルダーを追加するダイアログボックスが開かれます（*下のスクリーンショット*）。



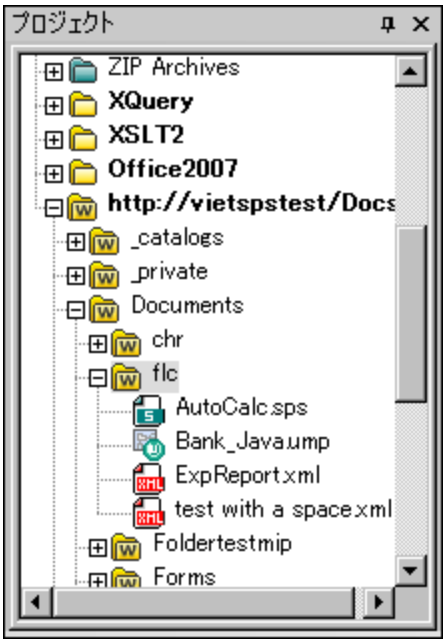
2. サーバーURL フィールドをクリックして、サーバーURL を入力します。サーバーがMicrosoft® SharePoint® Server の場合、このオプションをチェックします。サーバーのこの型のファイルと作業するための詳細に関しては、*Microsoft® SharePoint® Server* 下のセクションのフォルダーを参照してください。
3. サーバーがパスワードにより保護されている場合、ユーザーID とパスワードをユーザーとパスワードフィールドに入力してください。
4. **参照** をクリックして、サーバーに接続し、使用することのできるフォルダーを確認してください。



5. プロジェクトウィンドウに追加したいフォルダーをクリックします。この操作により、「開く」ボタンが有効になります。フォルダーの URL がファイル URL フィールドに表示されます。
6. 「開く」ボタンをクリックして、プロジェクトにフォルダーを追加します。



7. プラスアイコンをクリックして、フォルダー内部にあるコンテンツを確認することができます。



フォルダーのコンテンツをフィルタリング

フォルダーのコンテンツをフィルタリングするには、フォルダーを右クリックして、コンテキストメニューから「**プロパティ**」を選択します。表示されるプロパティダイアログにて、ファイル拡張子 フィールドをクリックし、表示させたいファイル種類の拡張子（例えば、XML やXSD ファイル）を入力し

ます。セミコロンによりファイルの種類を分けることができます(例: xml; xsd; sps)。プロジェクトウィンドウには、指定された拡張子を持つファイルだけが表示されるようになります。

フォルダーの検証と整形式のチェック

フォルダーに収められているファイルの整形式をチェック、または検証を行うには、フォルダーを選択して、[XML | **整形式をチェック**]または[XML | XML **の検証**]アイコンをクリックします(それぞれホットキー[**F7**] または[**F8**]が対応します)。フォルダーに表示されている全てのファイルがチェックされます。ファイルに不備があったり妥当でない場合、そのファイルがメインウィンドウで開かれ、編集を行うことができるようになります。エラーの修正を行い、再度チェックを行ってください。**Ctrl** キーを押下しながらファイルをクリックすることで、個別にファイルを選択することができ、「F7」または「F8」を押下することで、選択されたファイルがチェックされます。

プロジェクトフォルダーのコンテンツを更新

ファイルの追加など、ウェブフォルダー内のファイルは常に更新される可能性があります。フォルダーのビューを更新するには、外部フォルダーを右クリックして、コンテキストメニューから**最新の状態に更新**オプションを選択します。

フォルダーとファイルを削除

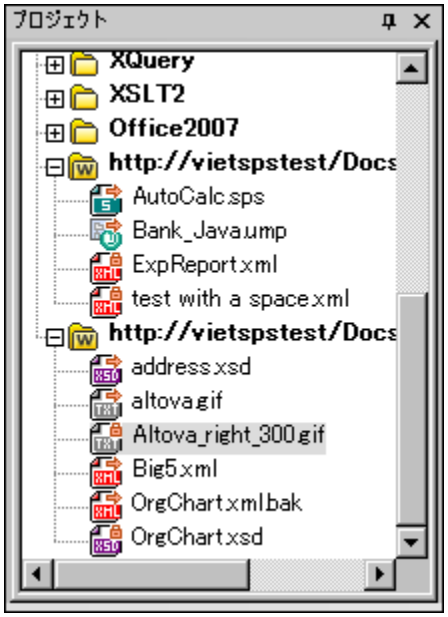
プロジェクトに追加されたのはウェブフォルダーだけであるため、プロジェクトから削除されるのもウェブフォルダーだけになります(つまりその中にあるファイルは削除されません)。ウェブフォルダーを削除するには、(i) フォルダーを右クリックして、「**削除**」を選択するか、(ii) フォルダーを選択して、「**削除**」キーを押下します。これでプロジェクトビューからフォルダーが削除されますが、ウェブサーバーにおけるファイル(やフォルダー)の削除は行われません。

メモ: 単一のファイルを右クリックして、「**削除**」キーを押しても、プロジェクトウィンドウからファイルを削除することはできません。ファイルを削除するにはサーバーから物理的にファイルを削除し、外部フォルダーのコンテンツを更新して確認する必要があります。

Microsoft® SharePoint® Server 上のフォルダー

Microsoft(R) SharePoint(R) Server 上にあるフォルダーがプロジェクトに追加されると、プロジェクトウィンドウ内に表示されているファイルのコンテキストメニューから、フォルダー内のファイルをチェックインまたはチェックアウトすることができるようになります。これらのコマンドにアクセスするには、目的のファイルを右クリックし、目的のコマンド(**チェックアウト**、**チェックイン**、**チェックアウトの取り消し**)を選択します。

ユーザーID ならびにパスワードはプロジェクト内部にある各フォルダーのプロパティ 808 に保存することができ、サーバーにアクセスするたびに、認証を行う必要が無くなります。



プロジェクトウィンドウ（上のスクリーンショット）では、ファイルのチェックイン/チェックアウトステータスを表すシンボルがファイルアイコンに表示されます。ファイルアイコンの説明を以下に示します：

	チェックイン状態。チェックアウト可能。
	他のユーザーによりチェックアウトされた状態。チェックアウト不可能。
	ローカルにチェックアウト済み。編集してチェックイン可能。

以下の点に注意してください：

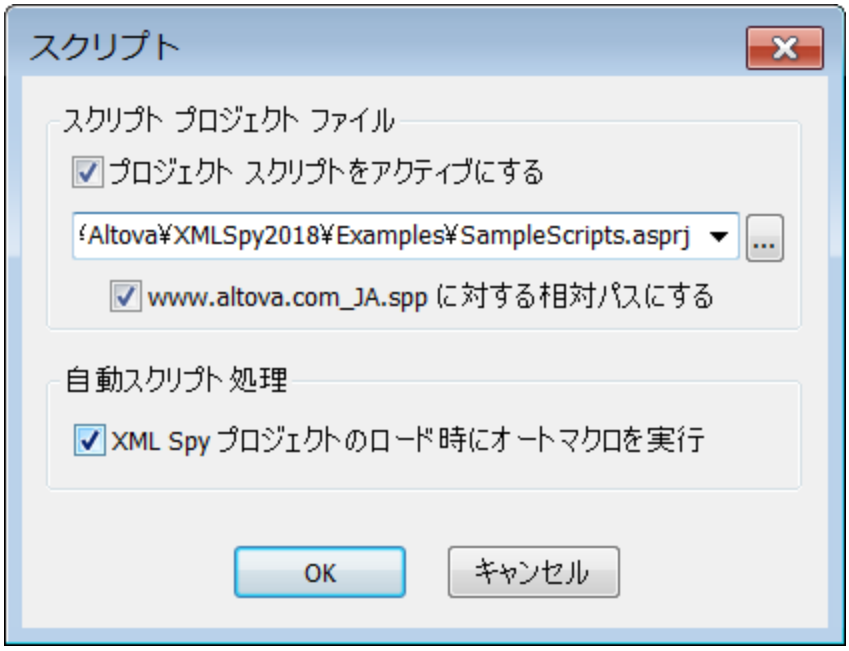
- ファイルのチェックアウトを行うと、Altova アプリケーション上でファイルの編集を行い、「**ファイル | 保存**（Ctrl+S）」により保存を行うことができます。
- 編集されたファイルのチェックインを行うには、プロジェクトウィンドウからアクセスすることができるコンテキストメニューか、アプリケーションのメインウィンドウにあるファイルのタブを右クリックすることで表示されるコンテキストメニューから、チェックインコマンドを選択します。

- 他のユーザーによりファイルがチェックアウトされている場合、そのファイルをチェックアウトすることはできません。
- （あなた自身により）ファイルがローカルにチェックアウトされている場合、コンテキストメニューからチェックアウトの取り消しを選択することで、チェックアウトを取り消すことができます。これでファイルが変更されないままサーバー側に返されることになります。
- ある Altova アプリケーションでファイルをチェックアウトしている場合、他の Altova アプリケーションにてそのファイルをチェックアウトすることはできません。ファイルは既にチェックアウトされている状態であると認識されます。Microsoft(R) SharePoint(R) Server をサポートしている Altova 製品が、この状態で行えるコマンドは、「**チェックインとチェックアウトの取り消し**」だけです。

24.3.15 スクリプトの設定

スクリプトプロジェクトは、XMLSpy プロジェクトに以下のとおり割り当てられます:

1. XMLSpy GUI 内で、必要な アプリケーションプロジェクトを開きます。
2. メニューコマンド「**プロジェクト | スクリプト設定**」を選択すると、スクリプト ダイアログが開かれます（*下のスクリーンショット*）。



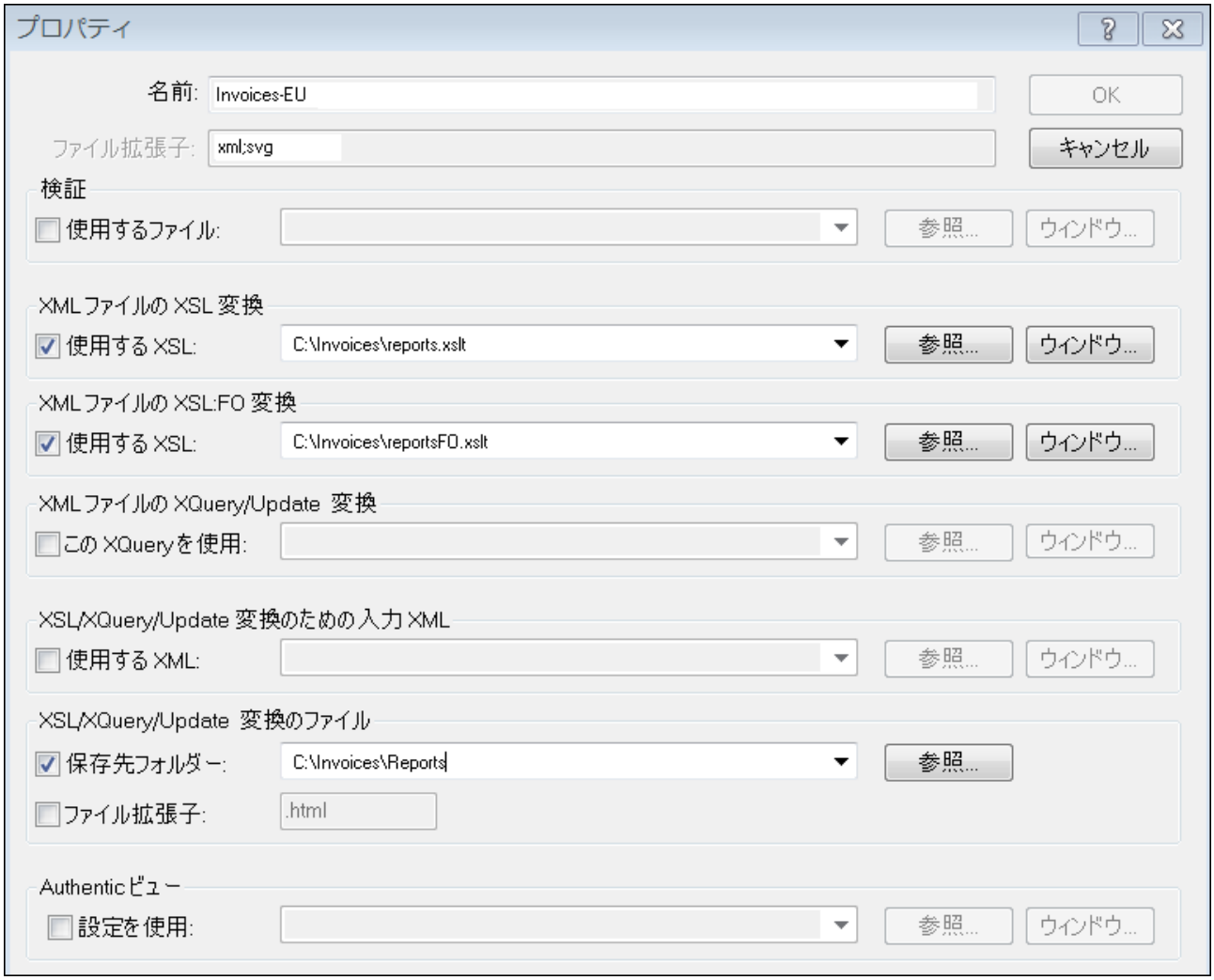
3. *プロジェクト スクリプトの有効化* チェックボックスをチェックし、必要な スクリプトプロジェクト（`.asprj` ファイル）を選択します。XMLSpy プロジェクトがロードされる時に、自動マクロを実行する場合、自動マクロを実行するチェックボックスをチェックします。
4. 「OK」をクリックして完了します。

メモ: XMLSpy プロジェクトのスクリプトプロジェクトを無効化（割り当てを解除）するには、*プロジェクト スクリプトを有効化するチェックボックスのチェック*を解除します。

24.3.16 プロパティ

「**プロジェクト | プロジェクトプロパティ**」コマンドは、アクティブなプロパティのプロパティダイアログを開きます（*下のスクリーンショット*）。（プロジェクトフォルダー自身をクリックすることとは異なり）プロジェクトウィンドウ内のフォルダーを右クリックし、「**プロパティ**」を選択すると、そのフォルダーのプロパティダイアログが開かれます。ダイアログ設定については下で説明されています。

メモ プロジェクトファイルがソース管理下にある場合、プロンプトが表示され、プロジェクト（`.spp`）ファイル。をチェックアウトするかが問われます。設定を編集し、保存するには、「OK」をクリックします。



設定

ファイル拡張子
ファイル拡張子 設定は、個別のフォルダーのために有効化されており、プロジェクトフォルダーのためには有効化されていません。ファイルがプロジェクトに追加されると、定義されているファイル拡張子を持つフォルダーに追加されます。`MyReport.xml` という名前のファイルがプロジェクトに追加されたと仮定します。(上のスクリーンショット内で示されるとおり) `.xml` ファイル拡張子が `Invoices-EU` フォルダーで設定されている場合、`MyReport.xml` が `Invoices-EU` フォルダーに追加されます。XML ファイルに追加するフォルダーが 1 つ以上存在する場合、個別の XML ファイルを(プロジェクトではなく)直接フォルダーに追加します。

外部フォルダーのためのユーザー ID とパスワード
(外部 Web フォルダーを含む) 外部フォルダーでは、サーバーへのアクセスに必要な可能性のあるユーザー ID とパスワードを保存することができます。

検証
現在のフォルダー(または、プロパティがプロジェクトの場合はプロジェクト全体)、DTD、XML スキーマ、または JSON スキーマ[537] 内で使用される 検証[841] ファイル。

XML ファイルの XSL 変換
XSLT スタイルシートはフォルダー内での XML ファイルの XSLT 変換[888] のために使用されます。

XML ファイルの XSL-FO 変換
XSLT スタイルシートは、フォルダー内の XSL-FO での XML ファイル変換のために使用されます。

XML ファイルの XQuery/Update 変換
フォルダー内の XML ファイルの XQuery または XQuery Update 実行のために使用される XQuery または XQuery Update ファイル。

XML ファイルの XSL/XQuery/Update 変換のための XML 入力
XSLT 変換のために入力として使用される XML ファイル、またはフォルダー内で対応する XSLT、XQuery または XQuery Update ファイルを持つ XQuery/XQuery Update 実行

XSL/XQuery/Update 変換のための出力ファイル
変換の保存先ディレクトリと任意の結果ドキュメントのファイル拡張子。

XULE 実行
XMLSpy アプリケーションウィンドウ内でアクティブな XULE ドキュメントを処理するための XBRL インスタンスファイル

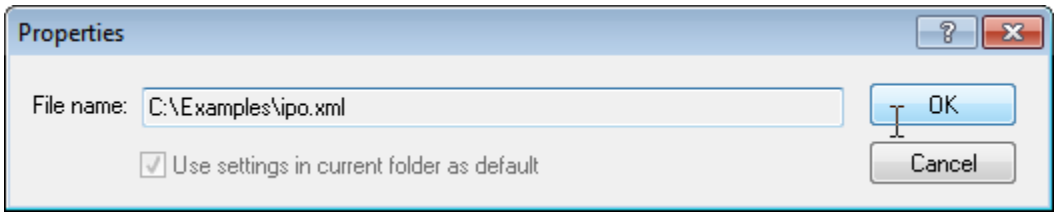
Authentic ビュー
*構成の使用*は、フォルダー内の XML ファイルの Authentic View 表示のために使用される StyleVision Power Stylesheet (SPS ファイル) を指定します。XML ファイルは、SPS のために使用される同じスキーマに対して有効である必要があります。

プロジェクトプロパティの注意点

以下の点と優先事項に注意してください:

- 変換、または、XSLT/XQuery 変換がプロジェクトフォルダーコンテキストメニューにより実行される場合、このダイアログで指定される検証 または変換 ファイルに XML ファイル内の割り当てより優先順位が与えられます。更に、個別のプロジェクトフォルダーのために指定されている設定に、祖先フォルダーのために指定されている設定より優先順位が与えられます。
- プロジェクトの複数フォルダー内で 1 つのファイルが存在する場合、異なるフォルダー内の異なる検証または変換 ファイルは、異なるフォルダーに割り当てられ、プロジェクトの外部でファイルが処理される場合、割り当てを設定することができます。これらの設定を指定する方法は以下のとおりです: 割り当てを希望するプロジェクトフォルダー内のファイルを検索します。プロジェクトフォルダー内のプロジェクトを右クリックして、「**プロパティ**」を選択します。(*下のスクリーンショット*) 表示されるダイアログ内で *「現在のフォルダ内の設定をデフォルトとして使用」*を選択します。(現在のフォルダーはファイルが存在するプロジェクトフォルダーです。) オプションが無効化された場合、現在のフォルダーの設定は、既にデフォルトの設定として使用されることを意味します。デフォルトではないプロジェクトフォルダー内のファイルインスタンスを選択すると、オプションが有効化され、デフォルトの設定をこのフォルダーの設定に切り替えることができます。ファイルにローカルの割り当て (ファイル自身内の割り当て) が存在する場合、ローカルの割り当てが使用され、デフォルトのフォルダー設定は無視されます。

(C) 2015-2021 Altova GmbH



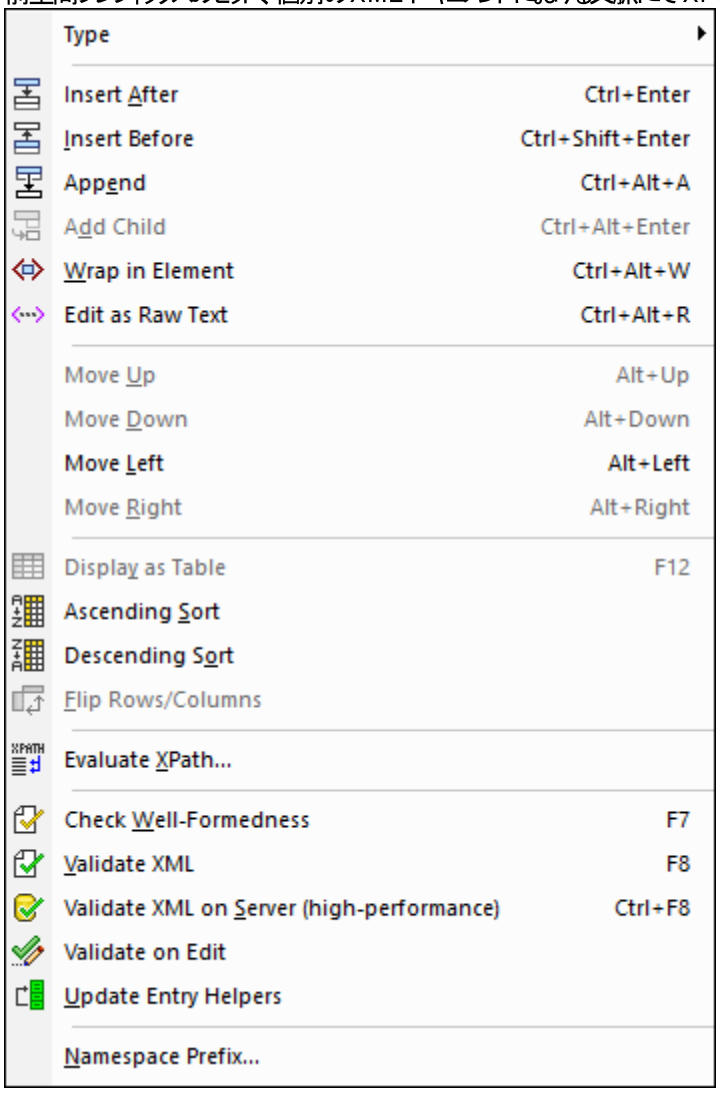
24.3.17　最近使用されたプロジェクト

このコマンドは最近使用されたプロジェクトを9つ表示し、それぞれのファイルへの素早いアクセスを提供します。

XMLSpy を起動すると、最後に使用されていたプロジェクト 1015 を自動的に開くこともできます(「**ツール | オプション | ファイル**」セクションにて、プログラム起動時に最後のプロジェクトを開く)。

24.4 XML メニュー

XML メニューには、XML ドキュメントの処理を行う際に使用されるコマンドが収められています。要素の挿入や追加、要素構造の変更、名前空間プレフィックスのセット、個別の XML ドキュメントにおける文脈にて XPath の評価を行うことができます。



最も頻繁に使用される XML 処理には、ドキュメントの整形式チェック 839 や、XML ドキュメントの検証 841 も含まれます。これらの処理に対応したコマンドはこのメニューに含まれます。

24.4.1 挿入

「XML | 挿入」コマンドは、(全てのビューで有効になってはいますが) グリッドビューでのみ使用することができます。サブメニューからは、以下にあるアイテムの挿入を行うことができます(以下のスクリーンショットを参照):

- XML 宣言ならびに XML ドキュメント内のノード型(属性、要素、テキスト、CDATA、コメント、処理命令)
- DOCTYPE 宣言ならびに XML ドキュメント内における外部 DTD 宣言

- DTD ドキュメント内の DTD 宣言（ELEMENT、ATTLIST、ENTITY、そして NOTATION）ならびに XML ドキュメントの DTD 宣言

24.4.1.1 属性の挿入

Ctrl+Shift+I

「XML | 挿入 | 属性」コマンドはグリッドビューでのみ使用でき、選択されたアイテムの前に新たな属性を挿入します。属性は、現在選択されているアイテムの親要素の直後、そして子要素の前に表示されるため、挿入された属性は現在選択されているアイテムから数行前の場所に挿入されることもあります。

24.4.1.2 要素の挿入

Ctrl+Shift+E

「XML | 挿入 | 要素」コマンドはグリッドビューでのみ使用でき、選択されたアイテムの前に新たな要素を挿入します。属性が選択されている場合、新たに作成される要素は、属性の親要素に含まれている最初の子要素の前に表示されます。

24.4.1.3 テキストの挿入

Ctrl+Shift+T

「XML | 挿入 | テキスト」コマンドはグリッドビューでのみ使用でき、選択されたアイテムの前に新たなテキストを挿入します。属性が選択されている場合、新たに挿入されるテキストは、属性の親要素に含まれている最初の子要素の前に表示されます。

24.4.1.4 CDATA の挿入

Ctrl+Shift+D

「XML | 挿入 | CDATA」コマンドはグリッドビューでのみ使用でき、選択されたアイテムの前に新たな CDATA ブロックを挿入します。属性が選択されている場合、新たに挿入される CDATA は、属性の親要素に含まれている最初の子要素の前に表示されます。

24.4.1.5 コメントの挿入

Ctrl+Shift+M

「XML | 挿入 | コメント」コマンドはグリッドビューでのみ使用でき、選択されたアイテムの前に新たなコメントを挿入します。属性が選択されている場合、新たに挿入されるコメントは、属性の親要素に含まれている最初の子要素の前に表示されます。

24.4.1.6 XML の挿入

「XML | 挿入 | XML」コマンドはグリッドビューでのみ使用でき、選択されたアイテムの前に新たな XML 宣言を挿入します。XML 宣言の子属性と、その値を挿入する必要があります。XML 宣言は以下のようになります:

```
<?xml version="1.0" encoding="UTF-8"?>
```

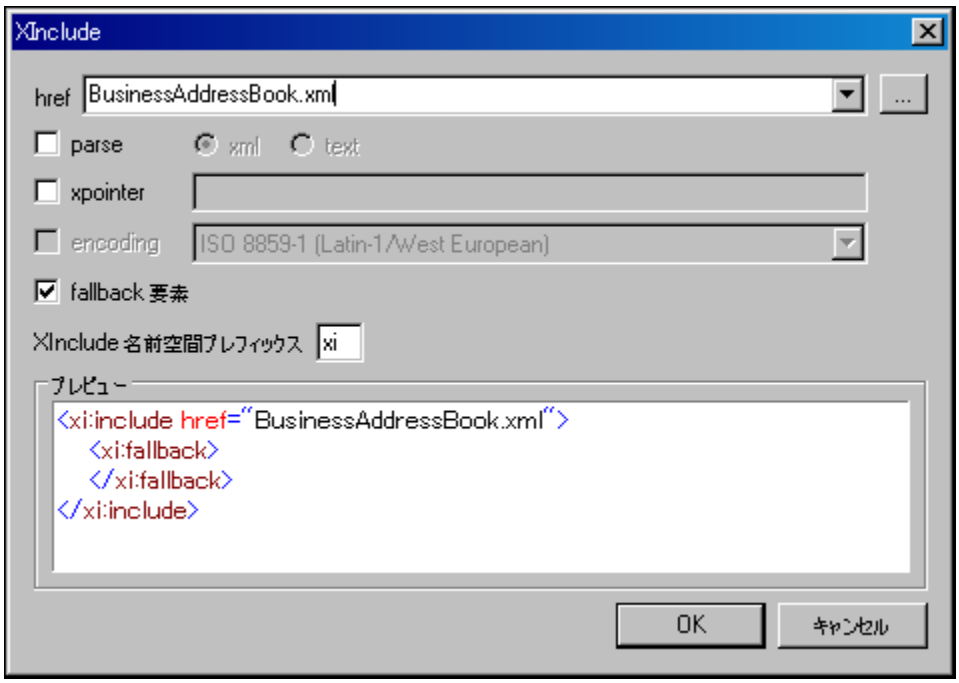
メモ: XML ドキュメントにはファイルのトップでしか XML 宣言を配置することができないため、このコマンドは XML 宣言が存在しないドキュメントの最上部でしか使用するべきではありません。

24.4.1.7 処理命令の挿入

「XML | 挿入 | 処理命令」コマンドはグリッドビューでのみ使用でき、選択されたアイテムの前に処理命令（PI）を挿入します。属性が選択されている場合、新たに挿入される処理命令（PI）は、属性の親要素に含まれている最初の子要素の前に表示されます。

24.4.1.8 XInclude の挿入

「XML | 挿入 | XInclude」コマンドはグリッドビューでのみ使用でき、選択されたアイテムの前に新たな XInclude 要素を挿入します。属性が選択されている場合、新たに挿入される XInclude 要素は、属性の親要素に含まれている最初の子要素の前に表示されます。このコマンドにより XInclude ダイアログが表示されます（以下のスクリーンショットを参照）。



インクルードされる XML ファイルを href テキストボックスに入力します（または、テキストボックスの右側にある… （参照）ボタンをクリックすることで、ファイルの参照を行うこともできます）。ファイル名が href 属性の値として、XML ドキュメント内に入力されます。XInclude (xi:include) 要素の parse、xpointer や encoding 属性、そして xi:include の fallback 子要素も、このダイアログにて挿入することができます。対応するチェックボックスにチェックを入れ、必要な値を選択/入力してください。fallback 要素の場合は、チェックボックスにチェックを入れることで、空要素を挿入することができます。fallback 要素のコンテンツは、編集ビューにて追加する必要があります。

parse 属性により、インクルードされるドキュメントが XML または テキストとして解析（パース）されるかが決定されます（デフォルトでは XML

という値がセットされており、指定を行う必要はありません)。xpointer 属性は、href 属性により指定されるドキュメント内フラグメント(つまり、インクルードされるフラグメント)を特定します。encoding 属性ではインクルードされるドキュメントのエンコーディングが指定され、XMLSpy により、インクルードするドキュメントの内容がインクルード元のドキュメントエンコーディングに合う形で変換されます。インクルードされるドキュメントが見つからない場合は、xi:include 要素がfallback 子要素により置換えられます。

XInclude により2つのXML ドキュメントをインクルードするXML ドキュメントのサンプルを以下に示します:

```
<?xml version="1.0" encoding="UTF-16"?>
<AddressBook xsi:schemaLocation="http://www.altova.com/sv/myaddresses AddressBook.xsd"
       xmlns="http://www.altova.com/stylevision/tutorials/myaddresses"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:xi="http://www.w3.org/2001/XInclude">
       <xi:include href="BusinessAddressBook.xml"/>
       <xi:include href="PersonalAddressBook.xml"/>
</AddressBook>
```

XML ドキュメントがパースされると、上の例で2つある XInclude 要素が、href 属性により指定されたファイルの内容に置換えられます。

xml:base

XMLSpy のXML バリデーターによりXML ドキュメントが読み取られ、XInclude 名前空間内のinclude 要素(これ以降は xi:include と表記)が発見されると、この要素(xi:include)を、xi:include 要素のhref 属性により指定されたXML ドキュメントに置換えます。インクルードされたXML ドキュメントのドキュメント要素(ルート要素)や、XPointer により指定された要素にもxml:base 属性は含まれ、インクルードされたベースURI を保持します。(インクルードされたXML やツリーフラグメントを含む)結果 XML ドキュメントが、スキーマに従う形で妥当でなければならない場合、インクルードされたドキュメントのドキュメント要素(ルート要素)やツリーフラグメントのトップレベル要素を、xml:base 属性を許可するコンテンツモデルにより作成する必要があります。スキーマにより、xml:base 属性がこの要素に対して認められない場合、ドキュメントは妥当でないと評価されます。XMLSpy のスキーマビューにて要素コンテンツモデルのxml:base 属性を定義する方法は、ドキュメンテーションのスキーマビューセクション、xml: プレフィックス属性 220 にて記述されます。

XPointers

XMLSpy ではXInclude 内におけるXPointer がサポートされます。関連するW 3C 勧告は、XPointer Framework ならびに XPointer element() スキーマの2つです。XInclude 要素内のXPointer により、XML ドキュメント全体の代わりに、XML ドキュメント内部でインクルードする箇所を特定することができます。XPointer は、以下にあるように、XInclude 要素内部で使用されます::

```
       <xi:include href="PersonalAddressBook.xml" xpointer="element(usa)"/>
       <xi:include href="BusinessAddressBook.xml" xpointer="element(/1/1)"/>
       <xi:include href="BobsAddressBook.xml" xpointer="element(usa/3/1)"/>
       <xi:include href="PatsAddressBook.xml" xpointer="element(usa)element(/1/1)"/>
```

XPointer のelement() スキーマでは、NCName や子シーケンスを使用することができます。

- 上の例にある最初のxi:include 要素のxpointer 属性ではusa というNCName を伴った要素スキーマが使用されています。XPointer Framework に従うと、このNCName によりusa というID を持つ要素が特定されます。
- 2つ目のxi:include では、element(/1/1) という値がxpointer 属性で使用されています。最初のステップでは、ドキュメントにある最初の子要素(つまりドキュメントが整形式であれば、ドキュメント要素)が特定されます。次のステップでは、最初のステップで特定された要素から見た最初の子要素が特定され、結果としてドキュメント要素にある最初の子要素がインクルードされます。
- 3番目のxi:include にあるxpointer 属性では、NCName と子シーケンスの両方が使用されています。XPointer により、usa という ID を持つ要素の3番目の子要素、その子要素から見た最初の子要素がインクルードされます。.
- 最初に使用したXPointer が期待通りに動作するか分からない場合、4番目の例として示されているxpointer="element(usa) element(/1/1)" という書式を使用することもできます。この方法では、たとえusa というID を持つ要素が存在しない場合でも、ドキュメント要素の最初の子要素が選択されます。これ以外にもバックアップとなる式を追加することができます。各 XPointer は、空白スペース以外で分けることはできず、以下のような書式に従う必要があります: xpointer="element(usa)element(addresses/1)

element(/1/1)"。

メモ: element() スキーマは qualified name でサポートされておらず、名前空間バインディングコンテキストは、element() スキーマにて使用されません。

24.4.1.9 DOCTYPE の挿入

「XML | 属性 | DOCTYPE」コマンドは XML ファイルを開いているグリッドビューで、最上位のノードが選択されている時のみ使用することができます。このコマンドにより DOCTYPE 宣言が XML ドキュメントのトップに追加されます。DOCTYPE の名前を入力する必要がありますが、名前はドキュメント要素のそれと同一でなければなりません。

DOCTYPE の名前を入力すると、DTD 内部サブセットで使用する宣言を入力することができます。

メモ:

- DOCTYPE 宣言は XML 宣言と XML ドキュメント要素の間にしか記述することができません。
- このコマンドの代わりに DTD の割り当て 852 コマンドを使用することで、外部 DTD ドキュメントを参照する DOCTYPE ステートメントを作成することもできます。

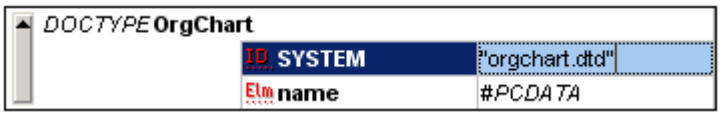
24.4.1.10 ExternalID の挿入

XML ファイル内の DOCTYPE 宣言には、DTD 宣言を含んでいる外部リソースに対する参照を含めることができます。このリソースには PUBLIC または SYSTEM 識別子により参照することができます。以下にその例を示します:

```
<!DOCTYPE doc_element_name PUBLIC "publicID" "systemID">
<!DOCTYPE doc_element_name SYSTEM "systemID">
```

SYSTEM 識別子は外部リソースを識別する URI です。PUBLIC 識別子は場所に依存するものではなく、外部リソースの場所を相互参照するために使用することができます。例えば、XMLSpy がインストールされると、広く使用されている DTD や XML スキーマが、catalog.xml という名前のファイル内に記述され、その実体が `C:\Program Files\Altova\Common2021\Schemas\` フォルダー以下に収められることになります。XML ドキュメント内の PUBLIC 識別子は、これらのファイル内に記述されている DTD の相互参照に使用することができます。

グリッドビューにて XML ファイル内の DOCTYPE アイテムの「子」アイテムが選択されていると、「XML | 挿入 | ExternalID」コマンドが有効になります。このコマンドにより、外部識別子（PUBLIC または SYSTEM）のグリッドビュー行が挿入されます。識別子の種類とその値を入力してください。



上に示されるスクリーンショットのグリッドビューに対応したテキストビューは、以下のようになります:

```
<!DOCTYPE OrgChart SYSTEM "orgchart.dtd" [
 <!ELEMENT name (#PCDATA)>
]>
```

メモ: External-ID の行は、選択されている DOCTYPE アイテムの子として追加したり、DOCTYPE アイテムの子アイテム（例: 上の例における name という ELEMENT 宣言）が選択されているときに挿入や追加することができます。

24.4.1.11 ELEMENT の挿入

「XML | 挿入 | ELEMENT」コマンドはグリッドビューでのみ使用でき、DTD ドキュメントや XML ドキュメントの DOCTYPE 宣言が選択されているときだけ選択可能になります。このコマンドにより、選択された宣言の前に ELEMENT 宣言が挿入されます。

24.4.1.12 ATTLIST の挿入

「XML | 挿入 | ATTLIST」コマンドはグリッドビューでのみ使用でき、DTD ドキュメントや XML ドキュメントの DOCTYPE 宣言が選択されているときだけ選択可能になります。このコマンドにより、選択された宣言の前に ATTLIST 宣言が挿入されます。

24.4.1.13 ENTITY の挿入

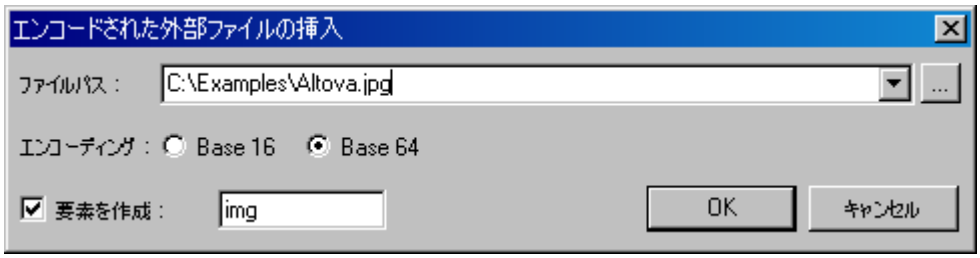
「XML | 挿入 | ENTITY」コマンドはグリッドビューでのみ使用でき、DTD ドキュメントや XML ドキュメントの DOCTYPE 宣言が選択されているときだけ選択可能になります。このコマンドにより、選択された宣言の前に ENTITY 宣言が挿入されます。

24.4.1.14 NOTATION の挿入

「XML | 挿入 | NOTATION」コマンドはグリッドビューでのみ使用でき、DTD ドキュメントやXML ドキュメントのDOCTYPE 宣言が選択されているときだけ選択可能になります。このコマンドにより、選択された宣言の前にNOTATION 宣言が挿入されます。

24.4.1.15 エンコードされた外部ファイルの挿入

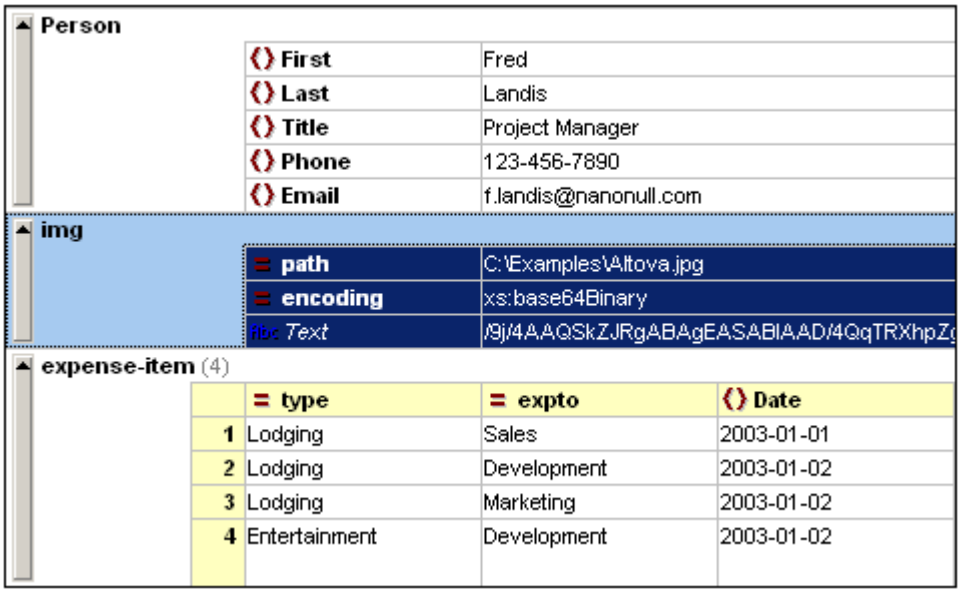
「**XML | 挿入 | エンコードされた外部ファイル**」コマンドはグリッドビューでのみ使用できます。イメージファイルといったバイナリファイルを、エンコードされた文字として挿入することができます。エンコードされた外部ファイルはグリッドビュー内で選択された要素の前に挿入されます。コマンドをクリックすると、エンコードされた外部ファイルの挿入ダイアログが表示されます（*以下のスクリーンショットを参照*）。ファイルへのパスを入力し、必要とするエンコードを選択し（周りの要素無しに）エンコードが現在選択されている要素の前にテキストとして挿入されるかを指定します。



エンコードならびに埋め込みが行われる外部ファイルの参照や名前の入力を行うことができます。Base-16 またはBase-64 エンコーディングを指定する必要があります。エンコードされたテキストを要素内に収めるには、要素の作成チェックボックスにチェックを入れて、*要素の作成*テキストボックスに使用する要素の名前を入力してください。*要素の作成*チェックボックスがチェックされていない場合、エンコードされたテキストはカーソル位置に直接挿入されます。

「OK」をクリックすることで選択されたファイルのエンコードされたテキストが囲まれた要素と共にカーソルの現在の選択個所の前に挿入されます。

エンコードされたファイルがグリッドビューにて表示されます（*以下のスクリーンショットにおけるハイライトされた部分*）。



テキストビューでは、以下のような内容がファイルに挿入されます。

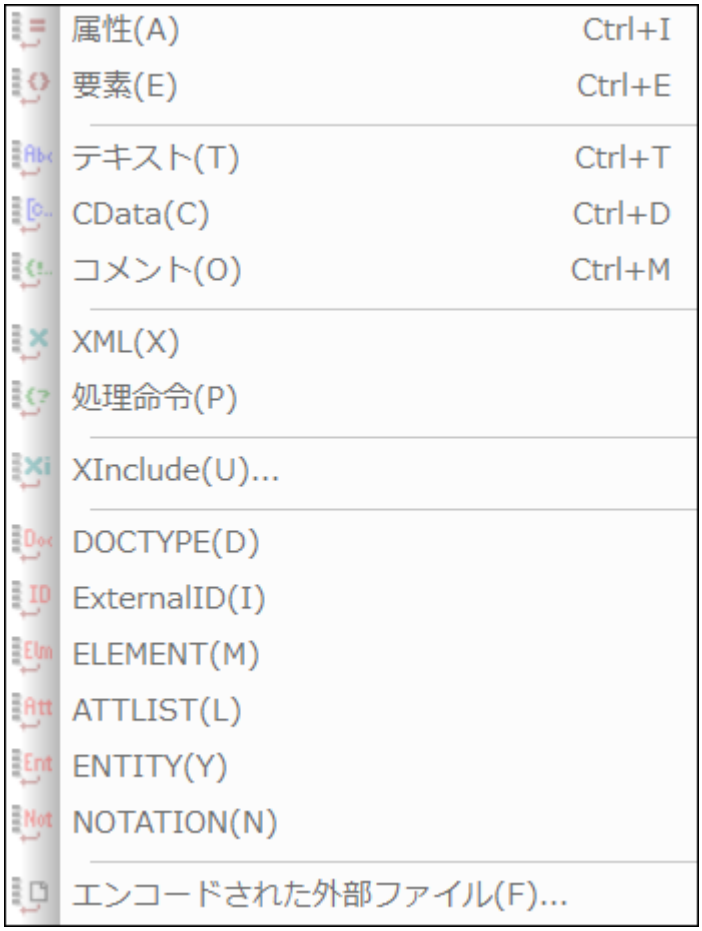
```
<img ext="jpg" encoding="xs:base64Binary">
  iVBORw0KGgoAAAANSUhEUgAAABAAAAAQAQMAAAAlPW0iAAAABlBMVEUAAAD/
  //+l2Z/dAAAAM0lEQVR4nGP4/5/h/1+G/58ZDrAz3D/McH8yw83NDDeNGe4U
  g9C9zwz3gVLMDA/A6P9/AFGGFyjOXZtQAAAAAElFTkSuQmCC
</img>
```

上の表示はJPG イメージファイルをテキストにエンコードしたものです。エンコードされたテキストの周りに `img` 要素が作成されます。

24.4.2 追加

「XML | 追加」コマンドは、(全てのビューで有効になってはいますが)グリッドビューでのみ使用することができます。サブメニューからは、以下にあるアイテムの追加を行うことができます(以下のスクリーンショットを参照):

- XML 宣言ならびにXML ドキュメント内のノード型(属性、要素、テキスト、CDATA、コメント、処理命令)
- DOCTYPE 宣言ならびにXML ドキュメント内における外部 DTD 宣言
- DTD ドキュメント内のDTD 宣言(ELEMENT、ATTLIST、ENTITY、そしてNOTATION)ならびにXML ドキュメントの DTD 宣言



24.4.2.1 属性の追加

Ctrl+I

「XML | 追加 | 属性」コマンドはグリッドビューでのみ使用でき、新たな属性を追加します。

24.4.2.2 要素の追加

Ctrl+E

「XML | 追加 | 要素」コマンドはグリッドビューでのみ使用でき、選択された要素の兄弟要素として、最後の位置に新たな要素ノードを追加します。属性ノードが選択されている場合、属性の親要素から見た子要素の最後に新たな要素が作成されます。

24.4.2.3 テキストの追加

Ctrl+T

「XML | 追加 | テキスト」コマンドはグリッドビューでのみ使用でき、選択された要素から見て最後の兄弟要素の後にテキストブロックが挿入されます。属性ノードが選択されている場合、属性の親要素から見た子要素の最後に新たなテキストブロックが作成されます。

24.4.2.4 CDATA の追加

Ctrl+D

「XML | 追加 | CDATA」コマンドはグリッドビューでのみ使用でき、(属性ノードを除く)選択された任意のノードから見た兄弟要素の最後にCDATA ノードを追加します。属性ノードが選択されている場合、属性の親要素から見た子要素の最後にCDATA セクションが追加されます。

24.4.2.5 コメントの追加

Ctrl+M

「XML | 追加 | コメント」コマンドはグリッドビューでのみ使用でき、(属性ノードを除く)選択された任意のノードから見た兄弟要素の最後にコメントノードを追加します。属性ノードが選択されている場合、属性の親要素から見た子要素の最後にコメントノードが追加されます。

24.4.2.6 XML の追加

「XML | 追加 | XML」コマンドは、新規 XML 宣言 `<?xml version="1.0" encoding="UTF-8"?>` を、ドキュメント最初のアイテムとして挿入します。

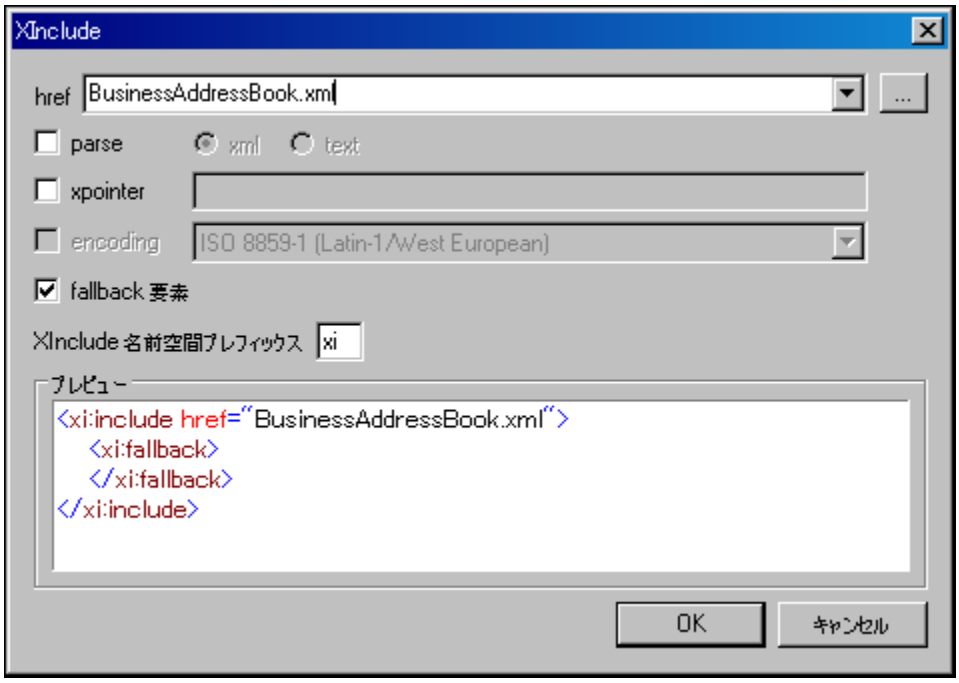
メモ: XML ドキュメントに含めることのできる XML 宣言は1つだけで、ファイルの最初に記述されなければなりません。

24.4.2.7 処理命令の追加

「XML | 追加 | 処理命令」コマンドはグリッドビューでのみ使用でき、(属性ノードを除く)選択された任意のノードから見た兄弟要素の最後に処理命令ノードを追加します。属性ノードが選択されている場合、属性の親要素から見た子要素の最後に処理命令ノードが追加されます。

24.4.2.8 XInclude の追加

「XML | 追加 | XInclude」コマンドはグリッドビューでのみ使用でき、(属性ノードを除く)選択された任意のノードから見た兄弟要素の最後にXInclude 要素を追加します。属性ノードが選択されている場合、属性の親要素から見た子要素の最後にXInclude 要素が追加されます。このコマンドによりXInclude ダイアログが表示されます(以下のスクリーンショットを参照)。



インクルードされるXML ファイルをhref テキストボックスに入力します(または、テキストボックスの右側にある… (参照)ボタンをクリックすることで、ファイルの参照を行うこともできます)。ファイル名がhref 属性の値として、XML ドキュメント内に入力されます。XInclude (xi:include) 要素のparse、xpointer やencoding 属性、そしてxi:include のfallback 子要素も、このダイアログにて挿入することができます。対応するチェックボックスにチェックを入れ、必要な値を選択/入力してください。fallback 要素の場合は、チェックボックスにチェックを入れることで、空要素を挿入することができます。fallback 要素のコンテンツは、編集ビューにて追加する必要があります。

parse 属性により、インクルードされるドキュメントがXML またはテキストとして解析（パース）されるかが決定されます（デフォルトではXML という値がセットされており、指定を行う必要はありません）。xpointer 属性は、href 属性により指定されるドキュメント内フラグメント（つまり、インクルードされるフラグメント）を特定します。encoding 属性ではインクルードされるドキュメントのエンコーディングが指定され、XMLSpy により、インクルードするドキュメントの内容がインクルード元のドキュメントエンコーディングに合う形で変換されます。インクルードされるドキュメントが見つからない場合は、xi:include 要素がfallback 子要素により置換えられます。

XInclude により2つのXML ドキュメントをインクルードするXML ドキュメントのサンプルを以下に示します:

```
<?xml version="1.0" encoding="UTF-16"?>
<AddressBook xsi:schemaLocation="http://www.altova.com/sv/myaddresses AddressBook.xsd"
       xmlns="http://www.altova.com/stylevision/tutorials/myaddresses"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:xi="http://www.w3.org/2001/XInclude">
       <xi:include href="BusinessAddressBook.xml"/>
       <xi:include href="PersonalAddressBook.xml"/>
</AddressBook>
```

XML ドキュメントがパースされると、上の例で2つあるXInclude 要素が、href 属性により指定されたファイルの内容に置換えられます。

xml:base

XMLSpy のXML バリデーターによりXML ドキュメントが読み取られ、XInclude 名前空間内のinclude 要素（これ以降はxi:include と表記）が発見されると、この要素（xi:include）を、xi:include 要素のhref 属性により指定されたXML ドキュメントに置換えます。インクルードされたXML ドキュメントのドキュメント要素（ルート要素）や、XPointer により指定された要素にもxml:base 属性は含まれ、インクルードされたベースURI を保持します。（インクルードされたXML やツリーフラグメントを含む）結果 XML ドキュメントが、スキーマに従う形で妥当でなければならない場合、インクルードされたドキュメントのドキュメント要素（ルート要素）やツリーフラグメントのトップレベル要素を、xml:base 属性を許可するコンテンツモデルにより作成する必要があります。スキーマにより、xml:base 属性がこの要素に対して認められない場合、ドキュメントは妥当でないと評価されます。XMLSpy のスキーマビューにて要素コンテンツモデルのxml:base 属性を定義する方法は、ドキュメンテーションのスキーマビューセクション、xml: プレフィックス属性 220 にて記述されます。

XPointers

XMLSpy ではXInclude 内におけるXPointer がサポートされます。関連するW 3C 勧告は、XPointer Framework ならびにXPointer element() スキーマの2つです。XInclude 要素内のXPointer により、XML ドキュメント全体の代わりに、XML ドキュメント内部でインクルードする箇所を特定することができます。XPointer は、以下にあるように、XInclude 要素内部で使用されます::

```
       <xi:include href="PersonalAddressBook.xml" xpointer="element(usa)"/>
       <xi:include href="BusinessAddressBook.xml" xpointer="element(/1/1)"/>
       <xi:include href="BobsAddressBook.xml" xpointer="element(usa/3/1)"/>
       <xi:include href="PatsAddressBook.xml" xpointer="element(usa)element(/1/1)"/>
```

XPointer のelement() スキーマでは、NCName や子シーケンスを使用することができます。

- 上の例にある最初のxi:include 要素のxpointer 属性ではusa というNCName を伴った要素スキーマが使用されています。XPointer Framework に従うと、このNCName によりusa というID を持つ要素が特定されます。
- 2つ目のxi:include では、element(/1/1) という値がxpointer 属性で使用されています。最初のステップでは、ドキュメントにある最初の子要素（つまりドキュメントが整形式であれば、ドキュメント要素）が特定されます。次のステップでは、最初のステップで特定された要素から見た最初の子要素が特定され、結果としてドキュメント要素にある最初の子要素がインクルードされます。
- 3番目のxi:include にあるxpointer 属性では、NCName と子シーケンスの両方が使用されています。XPointer により、usa という ID を持つ要素の3番目の子要素、その子要素から見た最初の子要素がインクルードされます。.
- 最初に使用したXPointer が期待通りに動作するか分からない場合、4番目の例として示されているxpointer="element(usa) element(/1/1)" という書式を使用することもできます。この方法では、たとえusa というID を持つ要素が存在しない場合でも、ドキュメント要素の最初の子要素が選択されます。これ以外にもバックアップとなる式を追加することができます。各 XPointer は、空白スペース

以外で分けることはできず、以下のような書式に従う必要があります: xpointer="element(usa)element(addresses/1) element(/1/1)"。

メモ: element() スキーマはqualified name でサポートされておらず、名前空間バインディングコンテキストは、element() スキーマにて使用されません。

24.4.2.9 DOCTYPE の追加

「XML | 追加 | DOCTYPE」コマンドはXML ファイルを開いているグリッドビューで、最上位のノードが選択されている時のみ使用することができます。このコマンドによりDOCTYPE 宣言がXML ドキュメントの上位に追加されます。DOCTYPE の名前を入力する必要がありますが、名前はドキュメント要素のそれと同一でなければなりません。

DOCTYPE の名前を入力すると、DTD 内部サブセットで使用する宣言を入力することができます。

メモ:

- DOCTYPE 宣言は XML 宣言と XML ドキュメント要素の間にしか記述することができません。
- 代わりに DTD の割り当て 852 コマンドを使用することで、外部 DTD ドキュメントを参照するDOCTYPE ステートメントを作成することもできます。

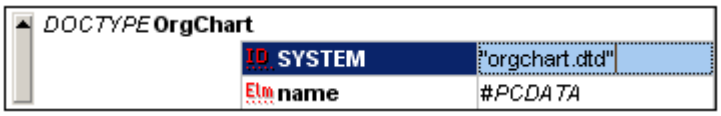
24.4.2.10 ExternalID の追加

XML ファイル内のDOCTYPE 宣言には、DTD 宣言を含んでいる外部リソースに対する参照を含めることができます。このリソースはPUBLIC またはSYSTEM 識別子により参照することができます。以下にその例を示します:

```
<!DOCTYPE doc_element_name PUBLIC "publicID" "systemID">
<!DOCTYPE doc_element_name SYSTEM "systemID">
```

SYSTEM 識別子は、外部リソースを識別するURI です。PUBLIC 識別子は場所に依存するものではなく、外部リソースの場所を相互参照するために使用することができます。例えば、XMLSpy XMLSpy がインストールされると、広く使用されているDTD やXML スキーマが、catalog.xml という名前のファイル内に記述され、その実体が`C:\Program Files\Altova\Common2021\Schemas\`フォルダー以下に収められることになります。XML ドキュメント内のPUBLIC 識別子は、これらのファイル内に記述されているDTD の相互参照に使用することができます。

グリッドビューにて XML ファイル内の DOCTYPE アイテムの「子」アイテムが選択されていると、「XML | 挿入 | ExternalID」コマンドが有効になります。このコマンドにより、外部識別子(PUBLIC または SYSTEM)のグリッドビュー行が挿入されます。識別子の種類とその値を入力してください。



上に示されるスクリーンショットのグリッドビューに対応したテキストビューは、以下のようになります:

```
<!DOCTYPE OrgChart SYSTEM "orgchart.dtd" [
 <!ELEMENT name (#PCDATA)>
]>
```

メモ: ExternalID の行は、選択されている DOCTYPE アイテムの子として追加したり、DOCTYPE アイテムの子アイテム(例: 上の例における name という ELEMENT 宣言)が選択されているときに挿入や追加することができます。

24.4.2.11 ELEMENT の追加

「XML | 追加 | ELEMENT」コマンドはグリッドビューでのみ使用でき、DTD ドキュメントや XML ドキュメントの DOCTYPE 宣言が選択されているときだけ選択可能になります。このコマンドにより、宣言のリストに ELEMENT 宣言が追加されます。

24.4.2.12 ATTLIST の追加

「XML | 追加 | ATTLIST」コマンドはグリッドビューでのみ使用でき、DTD ドキュメントや XML ドキュメントの DOCTYPE 宣言が選択されているときだけ選択可能になります。このコマンドにより、宣言のリストに ATTLIST 宣言が追加されます。

24.4.2.13 ENTITY の追加

「XML | 追加 | ENTITY」コマンドはグリッドビューでのみ使用でき、DTD ドキュメントや XML ドキュメントの DOCTYPE 宣言が選択されているときだけ選択可能になります。このコマンドにより、宣言のリストに ENTITY 宣言が追加されます。

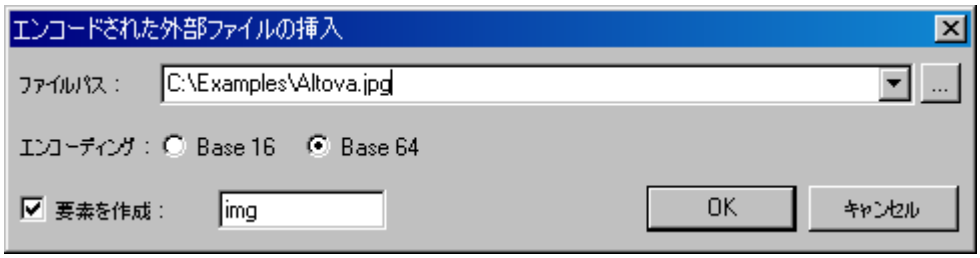
24.4.2.14 NOTATION の追加

「XML | 追加 | NOTATION」コマンドはグリッドビューでのみ使用でき、DTD ドキュメントやXML ドキュメントのDOCTYPE 宣言が選択されているときだけ選択可能になります。このコマンドにより、宣言のリストにNOTATION 宣言が追加されます。

24.4.2.15 エンコードされた外部ファイル

「XML | 追加 | エンコードされた外部ファイル」コマンドはグリッドビューでのみ使用できます。イメージファイルなどのバイナリファイルを、エンコードされた文字として追加することができます。エンコードされた外部ファイルがグリッドビューにおける選択対象の前に追加されます。

コマンドをクリックすると、エンコードされた外部ファイルの挿入ダイアログが表示されます（*下のスクリーンショットを参照*）。ファイルへのパスを入力し、必要とするエンコードを選択し、エンコードされたファイルが要素内に挿入されるかを指定します。



エンコードならびに埋め込みが行われる外部ファイルの参照や名前の入力を行うことができます。Base-16 またはBase-64 エンコーディングを指定する必要があります。エンコードされたテキストを要素内に収めるには、要素の作成チェックボックスにチェックを入れて、要素の作成テキストボックスに使用する要素の名前を入力してください。

「OK」をクリックすることで選択されたファイルのエンコードされたテキストがカーソル位置に指定されている場合囲まれた要素と共に挿入されます。

グリッドビュー内でエンコードされたファイルが追加されます。

24.4.3 子の追加

「XML | 子の追加」コマンドは、（全てのビューで有効になってはいますが）グリッドビューでのみ使用することができます。サブメニューからは、現在選択されている要素に対して、以下にある子アイテムを追加することができます（以下のスクリーンショットを参照）：

- XML 宣言ならびにXML ドキュメント内のノード型（属性、要素、テキスト、CDATA、コメント、処理命令）
- DOCTYPE 宣言ならびにXML ドキュメント内における外部 DTD 宣言
- DTD ドキュメント内のDTD 宣言（ELEMENT、ATTLIST、ENTITY、そしてNOTATION）ならびにXML ドキュメントの DTD 宣言

24.4.3.1 属性

Ctrl+Alt+I

「XML | 子の追加 | 属性」コマンドはグリッドビューで要素ノードが選択されている時のみ使用できます。新たな属性が選択された要素ノードの子として挿入されます。

24.4.3.2 要素

Ctrl+Alt+E

「XML | 子の追加 | 要素」コマンドはグリッドビューでのみ使用できます。選択されたノードの子として新たな要素を挿入します。

24.4.3.3 テキスト

Ctrl+Alt+T

「XML | **子の追加** | **テキスト**」コマンドはグリッドビューでのみ使用でき、選択されたアイテムの子として新たなテキストコンテンツを挿入します。

24.4.3.4 CDATA

Ctrl+Alt+D

「XML | **子の追加** | CDATA」コマンドはグリッドビューでのみ使用でき、選択されたアイテムの子として新たな CDATA セクションを挿入します。

24.4.3.5 コメント

Ctrl+Alt+M

「XML | **子の追加** | **コメント**」コマンドはグリッドビューでのみ使用でき、選択されたアイテムの子として新たなコメントノードを挿入します。

24.4.3.6 XML

「XML | **子の追加** | XML」コマンドはグリッドビューで**ファイルが空の時**のみ使用でき、ドキュメントにおける最初のアイテムとして XML 宣言 <?xml version="1.0" encoding="UTF-8"?> を挿入します。

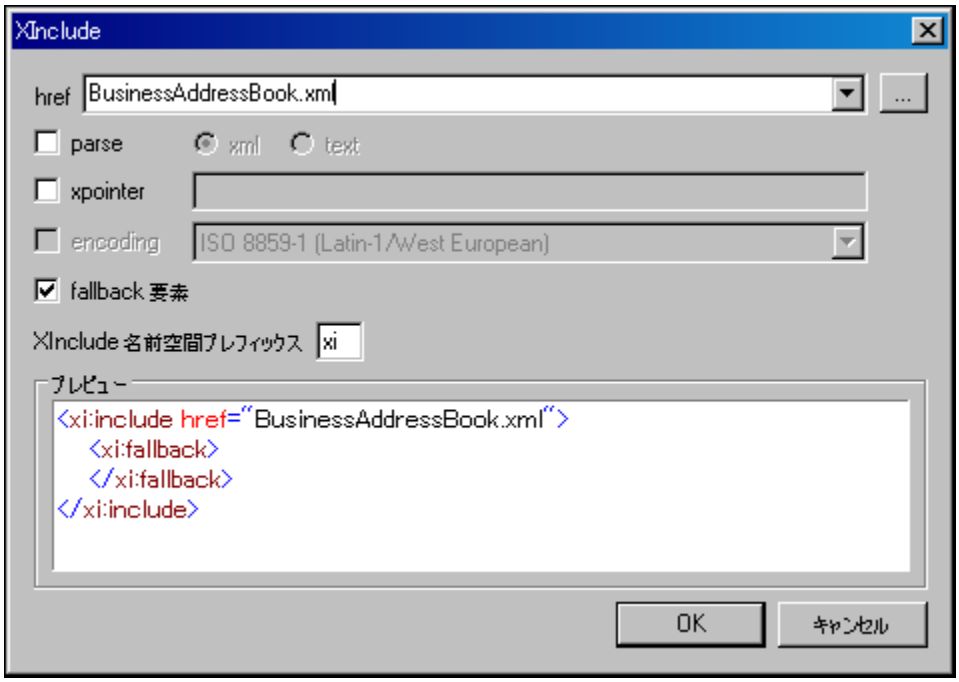
注意：XML ドキュメントに含めることのできる XML 宣言は1つだけで、ファイルの最初の方に記述されなければなりません。

24.4.3.7 処理命令

「XML | 子の追加 | 処理命令」コマンドはグリッドビューでのみ使用でき、選択されたアイテムの子として新たな処理命令（PI）を挿入します。

24.4.3.8 XInclude

「XML | 子の追加 | XInclude」コマンドはグリッドビューでのみ使用でき、選択されているアイテムの子として新たにXInclude 要素を挿入します。このコマンドによりXInclude ダイアログが表示されます（以下のスクリーンショットを参照）。



インクルードされるXML ファイルをhref テキストボックスに入力します（または、テキストボックスの右側にある… （参照）ボタンをクリックすることで、ファイルの参照を行うこともできます）。ファイル名がhref 属性の値として、XML ドキュメント内に入力されます。XInclude (xi:include) 要素のparse、xpointer やencoding 属性、そしてxi:include のfallback 子要素も、このダイアログにて挿入することができます。対応するチェックボックスにチェックを入れ、必要な値を選択/入力してください。fallback 要素の場合は、チェックボックスにチェックを入れることで、空要素を挿入することができます。fallback 要素のコンテンツは、編集ビューにて追加する必要があります。

parse 属性により、インクルードされるドキュメントがXML またはテキストとして解析（パース）されるかが決定されます（デフォルトではXML という値がセットされており、指定を行う必要はありません）。xpointer 属性は、href 属性により指定されるドキュメント内フラグメント（つま

り、インクルードされるフラグメント）を特定します。encoding 属性ではインクルードされるドキュメントのエンコーディングが指定され、XMLSpy により、インクルードするドキュメントの内容がインクルード元のドキュメントエンコーディングに合う形で変換されます。インクルードされるドキュメントが見つからない場合は、xi:include 要素がfallback 子要素により置換えられます。

XInclude により2つのXMLドキュメントをインクルードするXMLドキュメントのサンプルを以下に示します:

```
<?xml version="1.0" encoding="UTF-16"?>
<AddressBook xsi:schemaLocation="http://www.altova.com/sv/myaddresses AddressBook.xsd"
       xmlns="http://www.altova.com/stylevision/tutorials/myaddresses"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:xi="http://www.w3.org/2001/XInclude">
       <xi:include href="BusinessAddressBook.xml"/>
       <xi:include href="PersonalAddressBook.xml"/>
</AddressBook>
```

XMLドキュメントがパースされると、上の例で2つあるXInclude 要素が、href 属性により指定されたファイルの内容に置換えられます。

xml:base

XMLSpy のXMLバリデーターによりXMLドキュメントが読み取られ、XInclude 名前空間内のinclude 要素（これ以降はxi:include と表記）が発見されると、この要素（xi:include）を、xi:include 要素のhref 属性により指定されたXMLドキュメントに置換えます。インクルードされたXMLドキュメントのドキュメント要素（ルート要素）や、XPointer により指定された要素にもxml:base 属性は含まれ、インクルードされたベースURI を保持します。（インクルードされたXML やツリーフラグメントを含む）結果 XMLドキュメントが、スキーマに従う形で妥当でなければならない場合、インクルードされたドキュメントのドキュメント要素（ルート要素）やツリーフラグメントのトップレベル要素を、xml:base 属性を許可するコンテンツモデルにより作成する必要があります。スキーマにより、xml:base 属性がこの要素に対して認められない場合、ドキュメントは妥当でないと評価されます。XMLSpy のスキーマビューにて要素コンテンツモデルのxml:base 属性を定義する方法は、ドキュメンテーションのスキーマビューセクション、xml:プレフィックス属性 220 にて記述されます。

XPointers

XMLSpy ではXInclude 内におけるXPointer がサポートされます。関連するW 3C 勧告は、XPointer Framework ならびにXPointer element() スキーマの2つです。XInclude 要素内のXPointer により、XMLドキュメント全体の代わりに、XMLドキュメント内部でインクルードする箇所を特定することができます。XPointer は、以下にあるように、XInclude 要素内部で使用されます::

```
       <xi:include href="PersonalAddressBook.xml" xpointer="element(usa)"/>
       <xi:include href="BusinessAddressBook.xml" xpointer="element(/1/1)"/>
       <xi:include href="BobsAddressBook.xml" xpointer="element(usa/3/1)"/>
       <xi:include href="PatsAddressBook.xml" xpointer="element(usa)element(/1/1)"/>
```

XPointer のelement() スキーマでは、NCName や子シーケンスを使用することができます。

- 上の例にある最初のxi:include 要素のxpointer 属性ではusa というNCName を伴った要素スキーマが使用されています。XPointer Framework に従うと、このNCName によりusa というID を持つ要素が特定されます。
- 2つ目のxi:include では、element(/1/1) という値がxpointer 属性で使用されています。最初のステップでは、ドキュメントにある最初の子要素（つまりドキュメントが整形式であれば、ドキュメント要素）が特定されます。次のステップでは、最初のステップで特定された要素から見た最初の子要素が特定され、結果としてドキュメント要素にある最初の子要素がインクルードされます。
- 3番目のxi:include にあるxpointer 属性では、NCName と子シーケンスの両方が使用されています。XPointer により、usa というID を持つ要素の3番目の子要素、その子要素から見た最初の子要素がインクルードされます。.
- 最初に使用したXPointer が期待通りに動作するか分からない場合、4番目の例として示されているxpointer="element(usa)element(/1/1)" という書式を使用することもできます。この方法では、たとえusa というID を持つ要素が存在しない場合でも、ドキュメント要素の最初の子要素が選択されます。これ以外にもバックアップとなる式を追加することができます。各 XPointer は、空白スペース以外で分けることはできず、以下のような書式に従う必要があります: xpointer="element(usa)element(addresses/1)element(/1/1)"。

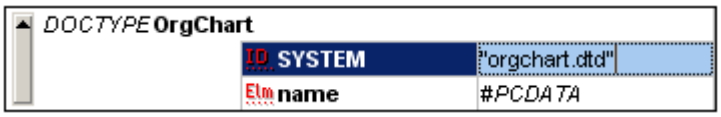
メモ: element() スキーマはqualified name でサポートされておらず、名前空間バインディングコンテキストは、element() スキーマにて使用されません。

24.4.3.9 DOCTYPE

「XML | 子の追加 | DOCTYPE」コマンドは空のドキュメントを表示しているグリッドビューでのみ使用できます。このコマンドによりXMLドキュメント内にDOCTYPE 宣言が挿入されます。DOCTYPE 宣言は内部DTD サブセットの宣言に使用することができます。

24.4.3.10 ExternalID

「XML | 子の追加 | ExternalID」コマンドはXML ファイルのDOCTYPE 宣言がグリッドビューで選択されている時のみ使用することができます。このコマンドにより、外部識別子(PUBLIC またはSYSTEM)のグリッドビュー行が挿入されます。識別子の種類とその値を入力してください。



上に示されるスクリーンショットのグリッドビューに対応したテキストビューは、以下のようになります:

```
<!DOCTYPE OrgChart SYSTEM "orgchart.dtd" [
 <!ELEMENT name (#PCDATA)>
]>
```

メモ: External-ID の行は、選択されているDOCTYPE アイテムの子として追加したり、DOCTYPE アイテムの子アイテム(例: 上の例におけるname というELEMENT 宣言)が選択されているときに挿入や追加することができます。

24.4.3.11 ELEMENT

「XML | 子の追加 | ELEMENT」コマンドはグリッドビューでのみ使用でき、DTD ドキュメントやXML ドキュメントのDOCTYPE 宣言が選択されている時だけ選択可能になります。このコマンドにより、宣言のリストにELEMENT 宣言が追加されます。

(C) 2015-2021 Altova GmbH

24.4.3.12 ATTLIST

「XML | 子の追加 | ATTLIST」コマンドはグリッドビューでのみ使用でき、DTD ドキュメントやXML ドキュメントのDOCTYPE 宣言が選択されている時だけ選択可能になります。このコマンドにより、宣言のリストにATTLIST 宣言が追加されます。

24.4.3.13 ENTITY

「XML | 子の追加 | ENTITY」コマンドはグリッドビューでのみ使用でき、DTD ドキュメントやXML ドキュメントのDOCTYPE 宣言が選択されている時だけ選択可能になります。このコマンドにより、宣言のリストにENTITY 宣言が追加されます。

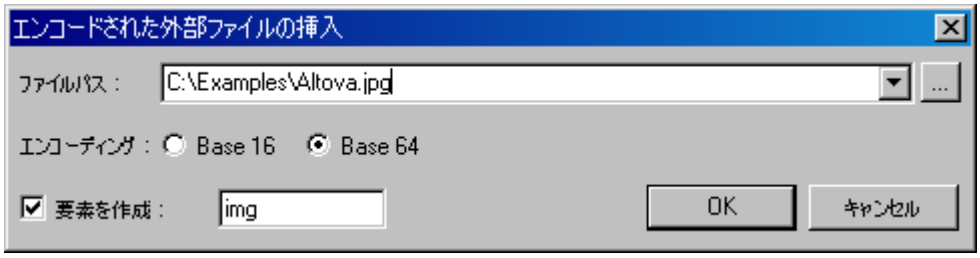
24.4.3.14 NOTATION

「XML | 子の追加 | NOTATION」コマンドはグリッドビューでのみ使用でき、DTD ドキュメントやXML ドキュメントのDOCTYPE 宣言が選択されている時だけ選択可能になります。このコマンドにより、宣言のリストにNOTATION 宣言が追加されます。

24.4.3.15 エンコードされた外部ファイル

「XML | 子の追加 | エンコードされた外部ファイル」コマンドはグリッドビューでのみ使用でき、イメージファイルといったバイナリファイルを、エンコードされた文字として追加することができます。エンコードされた外部ファイルがグリッドビューにおける選択対象の子として挿入されます。

コマンドをクリックすると、エンコードされた外部ファイルの挿入ダイアログが表示されます（以下のスクリーンショットを参照）。ファイルへのパスの入力とエンコードの選択を行い、エンコードされたファイルを要素の中に含めるか選択します。



「OK」をクリックすることで、（それが指定されている場合、要素に収められたかたちで）選択されたファイルのエンコードされたテキストがカーソル位置に挿入されます。

エンコードされたファイルが、追加された子としてグリッドビューに表示されます。

24.4.4 変換

「XML | 変換」コマンドにより、グリッドビューで選択されているアイテムを他のアイテム型に変換することができます。この処理はグリッドビューにて、子ノードを持たない単一のノードに対してのみ使用することができます。「カーソルを変換」コマンドにマウスオーバーするとサブメニューが表示され、選択されたアイテムから変換することができる種類のアイテムが表示されます（以下のスクリーンショットを参照）。

メモ: 選択された処理によりデータが失われるような場合（例えば属性をコメントに変換することで、属性名が失われます）、警告ダイアログボックスが表示されます。

24.4.4.1 属性

「XML | 変換 | 属性」コマンドにより、選択されたアイテムが属性に変換されます。

24.4.4.2 要素

「XML | 変換 | 要素」コマンドにより、選択されたアイテムが要素に変換されます。

24.4.4.3 テキスト

「XML | 変換 | テキスト」コマンドにより、選択されたアイテムがテキストコンテンツに変換されます。

24.4.4.4 CDATA

「XML | 変換 | CDATA」コマンドにより、選択されたアイテムがCDATA セグメントに変換されます。

24.4.4.5 コメント

「XML | 変換 | コメント」コマンドにより、選択されたアイテムがコメントに変換されます。

24.4.4.6 XML

「XML | 変換 | XML」コマンドにより、選択されたアイテムがXML 宣言に変換されます:

```
<?xml version="1.0" encoding="UTF-8"?>.
```

メモ: XML ドキュメントに含めることのできるXML 宣言は1つだけで、ファイルの最初に記述されなければなりません。

24.4.4.7 処理命令

「XML | 変換 | 処理命令」コマンドにより、選択されたアイテムが新たな処理命令（PI）に変換されます。

24.4.4.8 DOCTYPE

「XML | 変換 | DOCTYPE」コマンドにより、選択されたアイテムが(XML ファイル内の) DOCTYPE 宣言に変換されます。

メモ: DOCTYPE 宣言はXML 宣言とXML ドキュメント要素の間にしか記述することができません。

24.4.4.9 ExternalID

「XML | 変換 | ExternalID」コマンドにより、選択されたアイテムがDOCTYPE 宣言内のDTD 参照(PUBLIC またはSYSTEM 識別子)に変換されます。

24.4.4.10 ELEMENT

「XML | 変換 | ELEMENT」コマンドにより、選択されたアイテムがDOCTYPE 宣言または外部 DTD 内の要素型宣言に変換されます。

24.4.4.11 ATTLIST

「XML | 変換 | ATTLIST」コマンドにより、選択されたアイテムがDOCTYPE 宣言または外部 DTD 内の属性リスト宣言に変換されます。

24.4.4.12 ENTITY

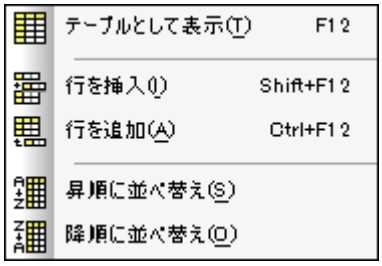
「XML | 変換 | ENTITY」コマンドにより、選択されたアイテムがDOCTYPE 宣言または外部 DTD の内エンティティ宣言に変換されます。

24.4.4.13 NOTATION

「XML | 変換 | NOTATION」コマンドにより、選択されたアイテムがDOCTYPE 宣言または外部 DTD 内の記法宣言(NOTATION) に変換されます。

24.4.5 テーブル

「**XML** | テーブル」コマンドは、(全てのビューで有効になってはいますが) グリッドビューでのみ使用することができます。サブメニューから、グリッドビューのテーブルの表示 250 に関連したコマンドにアクセスすることができます。



24.4.5.1 テーブルとして表示

テーブルとして表示 (F12) メニューコマンドとツールバーアイコン

XML | テーブルとして表示 コマンドにより、ドキュメント要素の標準的な グリッドビュー 141 と テーブルの表示 250 を切り替えることができます。テーブルビューでは、繰り返して表示される要素をテーブルとして表示することで、出現を行として、(コメントやCDATA セクション、処理命令などを含む) 子ノードをカラムとして扱うことができます。

24.4.5.2 行を挿入

Shift + F12

行またはセルが選択されると **行の挿入** コマンドは テーブルの表示 250 内で有効化されます。このコマンドにより、選択されている行の前に新たな行が挿入されます。新たな行はテーブル要素の出現に対応します。必須となっている子要素が新たな要素に対して作成されます

24.4.5.3 行を追加

Ctrl+F12

行またはセルが選択されると **行の付加** コマンドは テーブルの表示 250 内で有効化されます。これによりテーブルの最後に新たな行が挿入されます。新たな行は要素の出現に対応します。。必須となっている子要素が新たな要素に対して作成されます

24.4.5.4 昇順に並べ替え

行または セルが選択されると **昇順に並べ替え** コマンドは テーブルの表示 250 内で有効化されます。列のデータ型に従い列をアルファベット順または数値順に昇順に並べ替えられます。

24.4.5.5 降順に並べ替え

行または セルが選択されると **降順に並べ替え** コマンドは テーブルの表示 250 内で有効化されます。列のデータ型に従い列をアルファベット順または数値順に降順に並べ替えられます。

24.4.6 左へ移動

Ctrl+L

「XML | **左へ移動**」コマンドはグリッドビューでのみ使用することができます。このコマンドにより、選択されたノードが1つ左のレベルに移動され、親要素から見た子要素が兄弟要素になります。このコマンドは、「**レベル上げ**」コマンドと呼ばれることもあります。

24.4.7 右へ移動

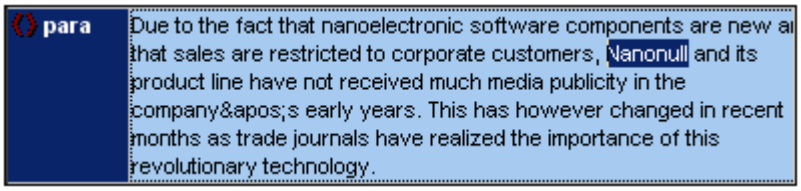
Ctrl+R

「XML | **右へ移動**」コマンドはグリッドビューでのみ使用することができます。このコマンドにより、選択されたノードが1つ右のレベルに移動され、ある要素が、1つ前の要素を親とする子要素になります。このコマンドは、「**レベル下げ**」コマンドと呼ばれることもあります。

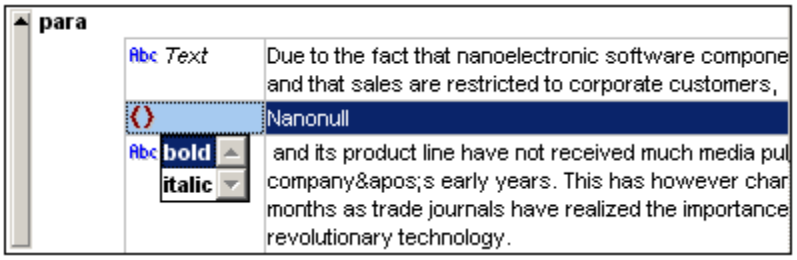
24.4.8 要素の中へ

「XML | **要素の中へ**」コマンドは、グリッドビューでのみ使用することができます。このコマンドにより、選択されたテキスト範囲を新たな要素の中に収めることができます。新たな要素が選択されたテキストの周りに作成されます。スキーマやDTD をベースにしたドキュメントを編集している場合、テキストを収めることができる妥当な要素名のリストが表示されます。

例えば、以下の例では para 要素内にある Nnonull というテキストがハイライトされています。



「XML | 要素の中へ」コマンドを選択すると、テキスト Nanonull が新たに作成されたインライン要素に収められ、要素の名前に使用される選択しとして bold と italic が表示されます。これらの要素は para の子としてスキーマでは定義されています。



ここで選択した結果が、新たに作成される要素の名前になります。それ以外にも要素に他の名前を入力することもできます。

24.4.9 XPath を評価

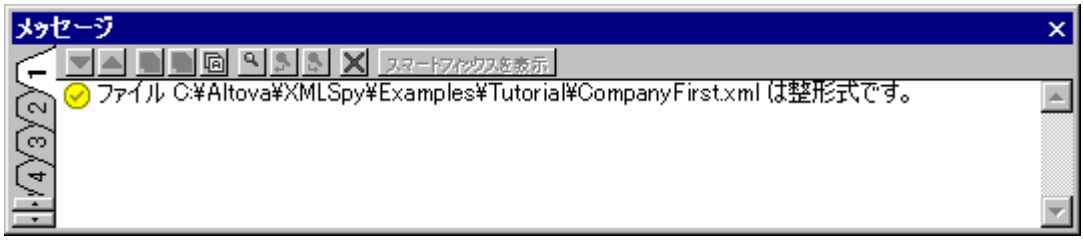
「XML | XPath を評価」コマンドにより、(出力ウィンドウが閉じられている場合は) 出力ウィンドウが開かれ、出力ウィンドウの XPath タブ 108 が有効になります。XPath タブでは、アクティブなドキュメントに対して XPath 条件式を評価することができ、その結果が出力ウィンドウに表示されます。

24.4.10 整形式のチェック

F7

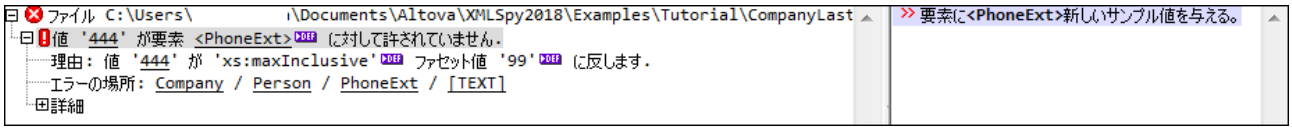
「XML | 整形式のチェック(F7)」コマンドにより、XML 1.0 仕様の定義に従い、アクティブなドキュメントの整形式をチェックすることができます。全ての XML ドキュメントは整形式でなければなりません。XMLSpy では、ドキュメントが開かれたときと保存します。またはテキストビューから他のビューへの切り替えが行われた時に整形式がチェックされます。このコマンドを使用することで、編集中はいつでも整形式のチェックを行うことができます。

整形式のチェックに成功すると、その旨のメッセージがメッセージウィンドウに表示されます(以下のスクリーンショットを参照)。



整形式チェックの結果、エラーが見つかった場合、エラーメッセージが表示されます（以下のスクリーンショットを参照）：

テキストビュー または グリッドビュー 内で整形式のチェックが行われた場合、スキーマ内の情報をベースにして、スマートフィックスのためのオプションが提案されます。スマートフィックス オプションのリストを閲覧するには、**スマートフィックスを表示** ボタンをクリックします（*上のスクリーンショット参照*）。スマートフィックスオプションを表示するペインがメッセージウィンドウ内に表示されます（*下のスクリーンショットを参照*）。メッセージウィンドウ内のエラーでは、1度に1つのエラーが表示されます。



スマートフィックス ペイン内で、「**フィックス + 検証**」ボタン または「**フィックス**」ボタンをクリックして提案されたスマートフィックスから選択します（*上のスクリーンショット参照*）。エラーは、スマートフィックスを使用して修正されます。または、スマートフィックスをダブルクリックすることもできます。このアクションは、ドロップダウンオプションリスト内で選択されたオプションに従い、エラーをフィックスするか、または、エラーをフィックスしドキュメントを検証します。「**フィックス + 検証**」コマンドは、固定されたエラー以外にも次のエラーが存在する場合はそのエラーをフィックスするため、とても役に立ちます。

スマートフィックス ペインを非表示にするには、「**スマートフィックスの非表示**」ボタンをクリックします（*上のスクリーンショット参照*）。

メモ メッセージウィンドウには9つのタブがあります。整形式チェックの結果はアクティブなタブに表示されます。そのため、タブ1にて整形式のチェックを行った後、タブ2に切り替えて、整形式チェックの結果を保持したまま、別のドキュメント検証を行うといったことができます（タブが切り替えられない場合、最初のタブに出力された情報が上書きされます）。

プロジェクト ウィンドウから検証を行う

[**検証**] コマンドをアクティブなプロジェクト内のファイル、フォルダー、ファイルのグループに適用することができます。（クリックして）プロジェクト ウィンドウ内で必要とされるファイルまたはフォルダーを選択します。[XML | Validate XMLの検証 841]をクリックします。サーバー上のXMLの検証（ハイパフォーマンス） 845 コマンドを使用することもできます。プロジェクト内の無効なファイルが開かれ、メインウィンドウ内でアクティブ化され、*ファイルが有効ではないことを知らせる*エラーメッセージが表示されます。

メモ　メッセージウィンドウには9つのタブがあります。整形式チェックの結果はアクティブなタブに表示されます。そのため、タブ1にて整形式のチェックを行った後、タブ2に切り替えて、整形式チェックの結果を保持したまま、別のドキュメント検証を行うといったことができます（タブが切り替えられない場合、最初のタブに出力された情報が上書きされます）。

通常問題のあるXMLドキュメントを保存することは許されていませんが、XMLSpy ではそれでも保存を行うことができます。作業を行なっているドキュメントの保存を（整形式で無い状態で）行い、後で作業を再開するような状況で、この機能を利用することができます。

メモ　**整形式のチェック**コマンドは、プロジェクトウィンドウ 105 でアクティブなファイルやフォルダー、ファイルのグループに対しても使用することができます。目的のアイテムをクリックし、整形式のチェックアイコンをクリックしてください。

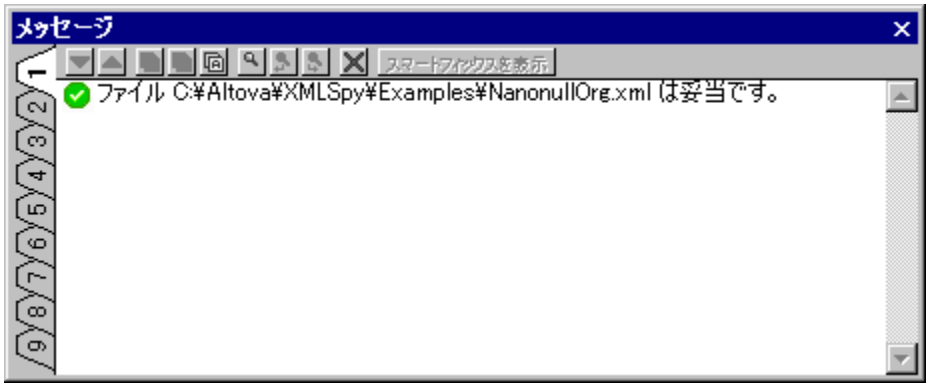
24.4.11　XML の検証

F8

「XML 検証(F8)」コマンドにより、DTD やXML スキーマ、またはその他のスキーマに対してXMLドキュメントの検証を行うことができます。テキストビューから他のビューへの切り替えが行われた際にも、検証は自動的に行われます。ファイルが開かれたり保存されたときに、自動的にドキュメントの検証が行われるよう指定することもできます（「**ツール | オプション | ファイル**」）。「**検証**」コマンドでは、検証が行われる前に整形式のチェックも行われます。そのため、「**検証**」コマンドを使用する前に整形式のチェック 839 を行う必要はありません。

メモ　データを編集中に検証するために編集中に検証 847 コマンドをオンに切り替えることができます。

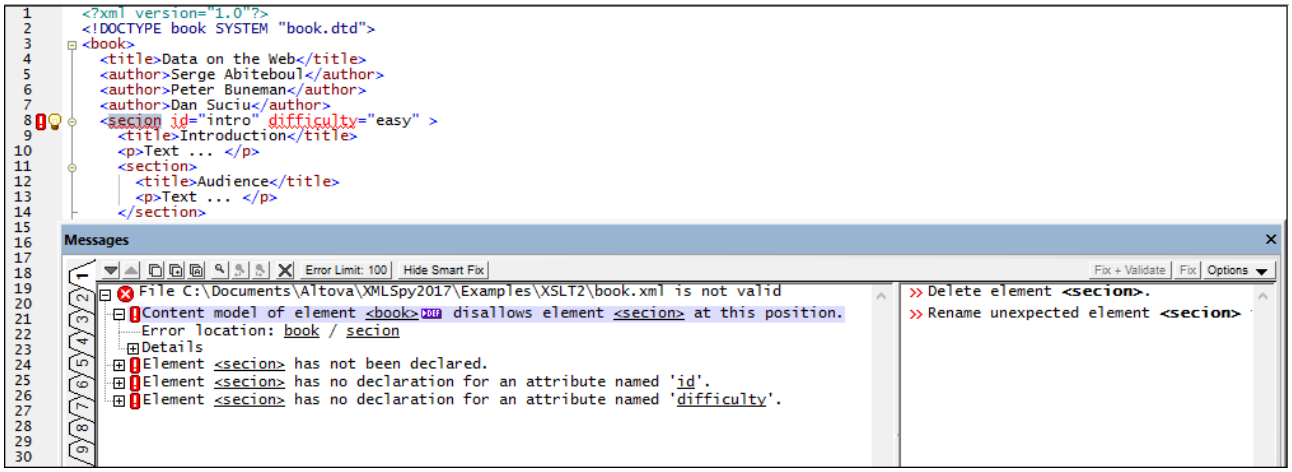
ドキュメントが有効な場合、メッセージウィンドウ内に検証の成功メッセージが表示されます。



そうでない場合、エラーの内容がメッセージウィンドウに表示されます（以下のスクリーンショットを参照）。リンクをクリックすることで、XML ファイル内のエラーが発見された箇所へジャンプすることができます。エラーメッセージの詳細と、XMLSpy のスマートフィックスを使用して検証エラーをフィックスする方法に関しては次のセクションを参照してください。

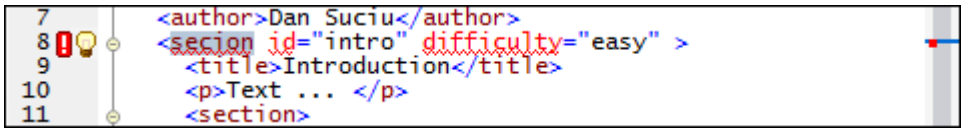
検証エラーとそのフィックス

「**テキストビュー**」または「**グリッドビュー**」にて検証が行われると、スキーマ内にある情報をベースに、修正の候補（スマートフィックス）が表示されます。スマートフィックスのオプションを参照するには、「**スマートフィックスを表示**」ボタンをクリックしてください（上のスクリーンショットを参照）。スマートフィックスの候補がメッセージウィンドウ内のペインに表示されます（以下のスクリーンショットを参照）。これらの提案を表示しない場合、「**スマートフィックスを非表示**」をクリックします。（一致しない開始と終了タグなどの）整形式のエラーが存在する場合、これらのエラーは検証エラーの前に表示されます。ですから（整形式 エラーがすべて修正された後）「**表示/非表示 スマートフィックス**」ボタンは検証エラーが発生すると有効化されます。

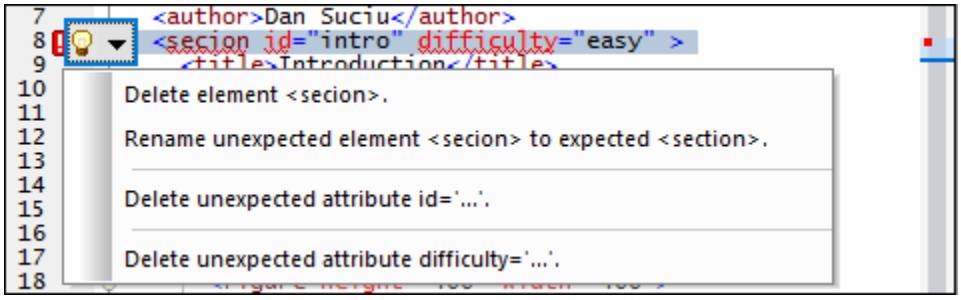


スマートフィックスを適用するには、以下を行います (i) ダブルクリックする (ii) 「フィックス」 または「フィックス + 検証」 オプションを選択してをクリックします (*下のスクリーンショットを参照*)。「フィックス + 検証」コマンドは、固定されたエラー以外にも次のエラーが存在する場合はそのエラーをフィックスします。

テキストビュー内では、2 つの検証エラーの追加インジケーターが存在します (*下のスクリーンショットを参照*): (i) 行番号マージン内の赤い感嘆符 アイコン、および、(ii) (ウィンドウの右の) スクロールバー内の赤い四角いマーカー。



感嘆符 アイコンの横の電球 アイコンは、スマートフィックスアイコンです (*上のスクリーンショット参照*)。マウスをポイントすると、エラーの原因の全てのスマートフィックスが表示されます (*下のスクリーンショットを参照*)。適用するスマートフィックスを選択します。



メモ ：出力ウィンドウには9つのタブがあります。検証の結果はアクティブなタブに表示されます。そのため、最初のタブにて整形式のチェックを行った後、2つ目のタブに切り替えて、別のドキュメント検証を行うといったことができます。2番目のドキュメントを検証するために、チェックを実行する前に、タブ2（または、タブ3）に切り替えます。タブを切り替えない場合、タブ1（またはアクティブな タブ）が最新の検証のフォーミュラにより上書きされます。

プロジェクトウィンドウからの検証

検証コマンドはアクティブなプロジェクト内部にあるファイル、フォルダー、またはファイルのグループに対しても適用することができます。目的のファイルやフォルダーをプロジェクトウィンドウにて（クリックにより）選択し、メニューオプションから「XML | XML 検証」を選択するか、「F8」を押下してください。プロジェクト内の妥当ではないファイルがメインウィンドウに表示され、ファイルが妥当でない旨のメッセージが表示されます。

XMLドキュメントの検証

XML ファイルを検証するには、XMLドキュメントをメインウィンドウにてアクティブな状態にした後、「XML | 検証」をクリックするか、「F8」キーを押下します。XML ファイルから参照されているスキーマに対してXMLドキュメントが検証されます。参照が無い場合、エラーメッセージがメッセージウィンドウに表示されます。XMLドキュメントが開かれている場合、スキーマもメモリ内に保管されます（DTD/スキーマメニュー以下にある フラッシュメモリキャッシュ 864 を参照ください）。

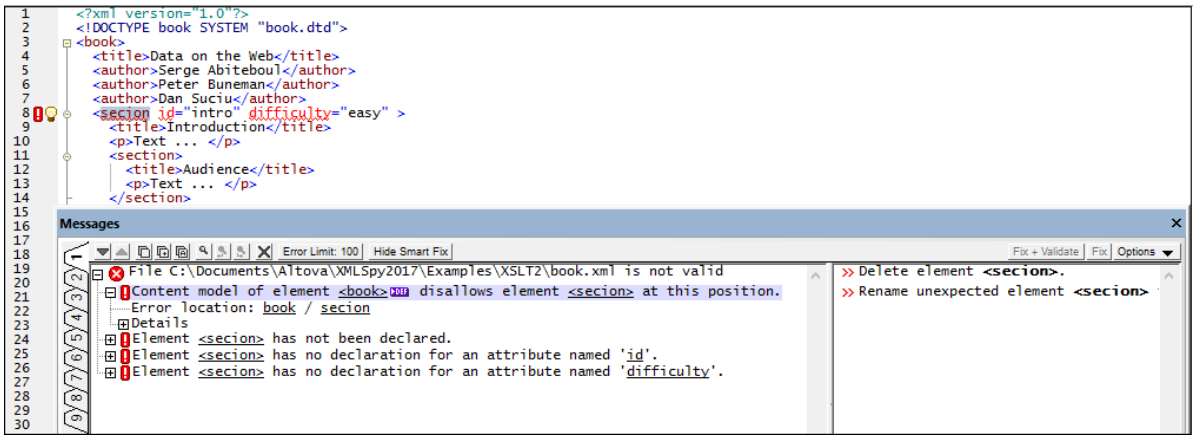
スキーマドキュメンテーション（DTD とXML スキーマ）の検証

XMLSpy では、DTD やXML スキーマといった広く使用されているスキーマがサポートされます。スキーマドキュメントの検証を行うには、ドキュメントをメインウィンドウにてアクティブにして、「XML | 検証」をクリックするか、「F8」 キーを押下します。

検証メッセージ

メッセージには以下の2種類があります:

- スキーマ（DTD または XML スキーマ）が妥当な場合、検証に成功した旨のメッセージがメッセージウィンドウに表示されます。
- スキーマが妥当で無い場合、エラーメッセージがメッセージウィンドウに表示されます（以下のスクリーンショットを参照）。



エラーメッセージは3つのパートから構成されます: 例えば、上のスクリーンショットでは、検証エラーを引き起こした可能性のある4つの理由についての詳細が報告されています。最初の理由は展開されており、3つの理由は折りたたまれています。それぞれの理由は3つの部分に分割されています:

1. エラーの説明: 説明には関連する宣言や定義へのリンクが含まれます。特定のスキーマ定義に移動し、ドキュメントが無効な理由を確認することができます。
2. エラーの原因と、エラーが発生したノードへのフルパス: パス内のノードをクリックすることで、ドキュメント内の該当するノードがハイライトされます。
3. エラーの詳細: 更に詳細なエラーの情報と、該当する仕様の関連する段落へのリンクが含まれています。

メモ 検証がテキストビューで行われると、メッセージウィンドウのリンクをクリックすることで、関連した宣言や定義がテキストビューにてハイライトされます。検証がスキーマビューで行われると、関連する宣言や定義が、デザインウィンドウやスキーマビュー、または関連する入力ヘルパーにてハイライトされます。

カタログ

XMLSpy ではOASIS XML カタログ機能のサブセットがサポートされます。カタログ機能により、一般的に使われているスキーマ(や、スタイルシートなどのファイル)が、XMLSpy によりローカルのユーザーフォルダーから取得されます。これにより全体的な処理スピードが向上し、(ネットワークに接続していない状態でも)ユーザーはオフラインで作業することができるようになり、(URI の変更はカタログファイル内部に限定されるので)ドキュメントのポータビリティーが向上します。XMLSpy のカタログ機能は以下のように動作します:

- XMLSpy により、検索を行うカタログファイルのリストが含まれる RootCatalog.xml というファイルがロードされます。RootCatalog.xml ファイルにて nextCatalog 要素を加えることで、検索を行ないたいファイルを増やすこともできます。
- RootCatalog.xml に収められたカタログファイルが検索され、カタログファイルで指定されたマッピングにより、URI が解決されます。エラーを起こす可能性があるため、マッピングの複製を行わないよう注意する必要があります。
- XMLSpy では2つのカタログファイルが用意されています。これらの動作方法については XMLSpy のカタログ 301 セクションを参照ください。
- XML ファイルの DOCTYPE ステートメントにおける PUBLIC ならびに SYSTEM 識別子はカタログの検索にて使用されます。広く使用されているスキーマでは、PUBLIC 識別子が既に定義されており、複数のコンピューターで XML ドキュメントが使用されている場合カタログファイル内部の URI を変更するだけで十分となります。

CustomCatalog.xml ファイル(または他のカスタムカタログファイル)を編集する場合、XMLSpy によるカタログが正しく行われるように、以下のサブセットだけを使用するようにしてください。サポートされているサブセットの各要素には xml:base 属性を含めることができ、その要素のベースURI を指定するのに使用することができます。

```
<catalog...>
...
```

```
<public publicId="PublicID of Resource" uri="URL of local file"/>
<system systemId="SystemID of Resource" uri="URL of local file"/>
<rewriteURI uriStartString="StartString of URI to rewrite" rewritePrefix="String to
replace StartString"/>
<rewriteSystem systemIdStartString="StartString of SystemID"
rewritePrefix="Replacement string to locate resource locally"/>
<uri name="filename" uri="URL of file identified by filename"/>
...
</catalog>
```

メモ

- %AltovaCommonFolder%\Schemas\schema 301 フォルダー内の catalog.xml ファイルには、古いバージョンの XML 仕様を実装する DTD への参照が含まれています。これらスキーマを、XML スキーマドキュメントの検証に対して使用するべきではありません。参照された DTD ファイルは、古い勧告に従ったドキュメントの編集時に使用される XMLSpy の入力ヘルパーに対してのみ利用されます。
- 特定のスキーマに対してカスタムファイル拡張を作成すると、CustomeCatalog.xml に1行追加することで、インテリジェントな編集をその拡張子を持ったファイルに対しても使用することができるようになります（例えば、.myhtml という拡張子により、HTML DTD を使った HTML ファイルの検証を行うことができます）。この例に従うと、<spy:fileExtHelper ext="myhtml" uri="schemas/xhtml/xhtml1-transitional.dtd"/> という要素を <catalog> 要素の子要素として追加することになります。この操作により、XHTML 1.0 Transitional DTD に従う形で、自動補完や入力ヘルパーといったインテリジェントな編集機能を使いながら、.myhtml ファイルの編集を XMLSpy で行うことができるようになります。
- カタログに関するより詳細な情報については、XML カタログ仕様を参照下さい。

RaptorXML 2021 を使用して検証を自動化する

RaptorXML は、XML 検証、XSLT 変換、とXQuery 変換のためのAltova のスタンドアロンアプリケーションです。Java プログラムと.NET 内のCOM インターフェイスを使用して、アプリケーションコマンドラインから使用することができます。RaptorXML を使用すると、検証のタスクを自動化することもできます。例えば、ドキュメントのセット上で検証を実行するためにRaptorXML を呼び出すバッチファイルを作成し、テキストファイルに出力を送ることができます。詳細に関しては、RaptorXML ドキュメンテーション を参照ください。

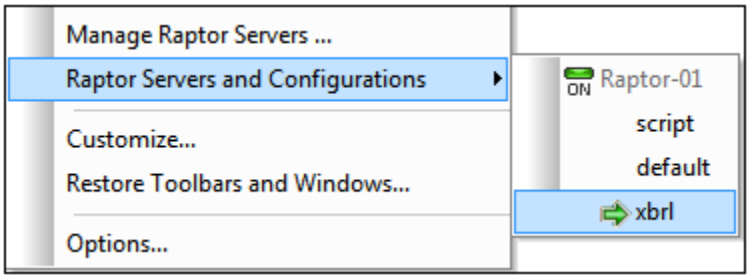
24.4.12 サーバー上の XML の検証（ハイパフォーマンス）

Ctrl+F8

XML | サーバー上の検証（ハイパフォーマンス）（Ctrl+F8）コマンドは、現在アクティブな RaptorXML Server 998 とアクティブな 構成 998 を使用してアクティブな XML ドキュメントを検証します。コマンドはすぐに検証を行い、メッセージウィンドウ内のフォーミュラを表示します。

メモ: 実際のパフォーマンスは、検証のためにRaptorXML Server により使用されるPC プロセッサコアにより異なります。使用されるコア数が多いほど、処理が速くなります。

複数のサーバーで複数の構成を定義した場合、サーバーとそのアクティブな構成として1つの構成を選択することができます。アクティブな構成は、以降の検証に使用されます。カーソルを「ツール | Raptor Servers および 構成」コマンドにポイントして、追加されたすべてのサーバーとその詳細が含まれるサブメニューから、を含む表示されるサブメニューが表示されます（下のスクリーンショット参照）。アクティブな構成と設定するサーバー構成を選択します。したのスクリーンショットでは、Raptor-01 という名前のサーバーの(緑色の矢印で示されている) xbrl 構成がアクティブな構成として選択されています。



サーバー上の XML の検証（ハイパフォーマンス）（Ctrl+F8）コマンドをプロジェクト 入力ヘルパー内で使用することができます。プロジェクト、フォルダー、または、ファイルを右クリックし、**サーバー上の XML の検証** を選択し、選択されたオブジェクト内の XML または XBRL データを検証します。

メモ: Raptor 検証はテキストビュー、グリッドビュー、および XBRL ビューで使用することができます。

24.4.13 WSDL ファイルの検証

W SDL ドキュメントは W SDL だけではなく XML でもあります。この結果 XML および W SDL として検証することができます。次のリストは XMLSpy の *Enterprise* と *Professional Editions* 内での W SDL 検証[841] の振る舞いに関する重要な情報が含まれています。

- *Professional Edition* は W SDL ファイルを XML ファイルとして扱い `http://schemas.xmlsoap.org/wsdl/` で定義されるスキーマに従い検証する単純なスキーマ検証を行います。
- *Enterprise Edition* では、*Professional Edition* を超えた XML 検証を行うことができます。`http://schemas.xmlsoap.org/wsdl/` に対する検証は行われません。代わりに `http://www.altova.com/specs_wsdl.html#_document-s` と独自のロジックを使用することで、W SDL の文脈に沿った更に高度な情報が提供されます。そのため、*Professional Edition* では妥当とされた W SDL ファイルが、*Enterprise Edition* では、そうでないと表示されることがあります（*以下の例を参照*）。
- `http://schemas.xmlsoap.org/wsdl/` と `http://www.altova.com/specs_wsdl.html#_document-s` には違いがあります。前者には W SDL 仕様で定義されている拡張要素（**extensibility elements**）の定義が含まれていません。この短所は公式な W 3C スキーマ内でエラーとして表示されます。この短所は（*Enterprise Edition* により使用されている）後者のスキーマで指摘されています。
- *Professional Edition* は `http://schemas.xmlsoap.org/wsdl/` を検証のために使用するため、拡張性要素は *Professional Edition* 内で無効とレポートされますが、（`http://www.altova.com/specs_wsdl.html#_document-s`）を使用する *Enterprise Edition* 内では有効とレポートされます。
- W 3C スキーマは W 3C 作業部会により提供されている公式のスキーマのため、発生するエラーに関しては、このスキーマが原因となるエラーに対しては Altova が対処することはできません。

例

以下に W SDL ファイルの一部を例として示します。このファイルでは要素 `getCityTimes` が宣言されています。この要素は、`getCityTimes` という間違った名前で参照されています。*Enterprise Edition* では、参照された要素が、ファイル内で既に宣言されているかをチェックしますが、*Professional Edition* にてこのチェックは行われません。ファイルの他の箇所が妥当であることを前提とすると、このファイルは *Professional Edition* にて妥当であると見なされ、*Enterprise Edition* では（`getCityTimes` がファイルのその他の場所にて宣言されていない場合）不正となります。

```
<s:element name="getCityTime">
```

```
   <s:complexType>
      <s:sequence>
         <s:element minOccurs="0" maxOccurs="1" name="city" type="s:string"/>
      </s:sequence>
   </s:complexType>
</s:element>
<s:element name="abc">
   <s:complexType>
      <s:sequence>
         <s:element ref="getCityTimes"/>
      </s:sequence>
   </s:complexType>
</s:element>
```

24.4.14　編集中に検証

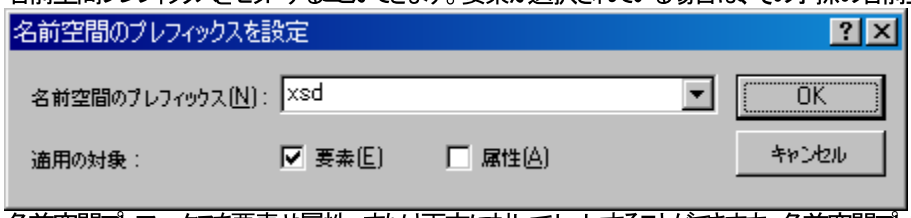
編集中に検証 コマンドは テキストビュー 122 、JSON グリッドビュー 485 、または Authentic ビューで入力中に検証を行う *編集中に検証* モードをオンとオフに切り替えます。コマンドのツールバーボタン または、オプションダイアログのファイル 1015 の *検証 > 編集中* オプションによりモードを切り替えることができます。

24.4.15　入力ヘルパーの更新

「XML | **入力ヘルパーを更新**」コマンドにより、使用されている DTD やスキーマを再ロードして、入力ヘルパーを更新します。開かれている XML ドキュメントがベースにしている XML スキーマや DTD を修正した場合、入力ヘルパーの更新を行って、スキーマにおける編集をインテリジェントな編集機能にも反映させるべきでしょう。

24.4.16　名前空間プレフィックス

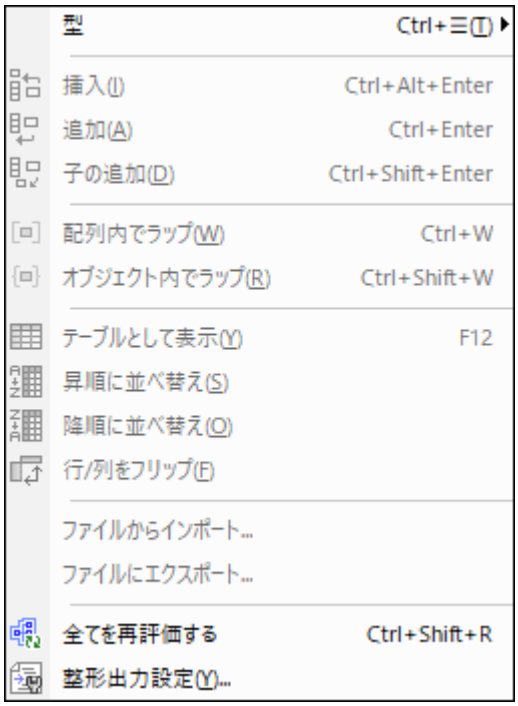
「XML | **名前空間プレフィックス...**」コマンドはグリッドビューでのみ使用することができ、ダイアログボックスから選択されている要素や属性の名前空間プレフィックスをセットすることができます。要素が選択されている場合は、その子孫の名前空間プレフィックスも変更されます。



名前空間プレフィックスを要素や属性、または両方に対してセットすることができます。名前空間プレフィックスは選択された要素や属性に、そして属性が選択されている場合はその子孫ノードに対しても、適用されます。

24.5 JSON メニュー

JSON メニューには、JSON ドキュメントの処理を行う際に使用されるコマンドが収められています。コマンドの多くは JSON グリッドビュー[475] と作業するために使用されます。コマンドが現在の箇所に適用できない場合、これらのコマンドは無効化されます。

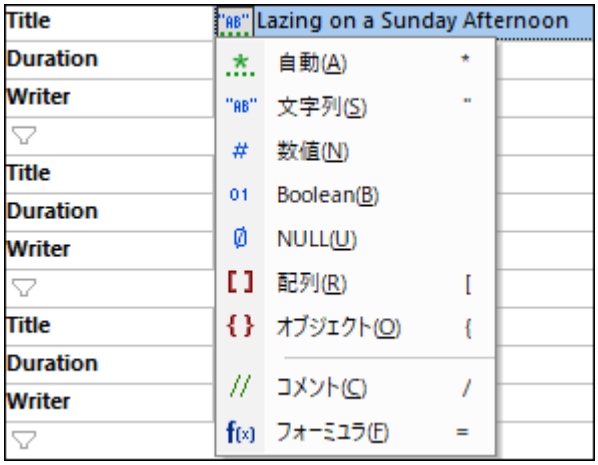


このメニューのコマンドはこのセクションのサブセクションで説明されています:

- 型[848]
- 子の挿入、付加、追加[849]
- 配列/オブジェクト内でラップ[849]
- 右から左への読み取り[849]
- 移動[850]
- コピー[850]
- 展開、折りたたみ[850]
- 昇順に並べ替え、降順に並べ替え[851]

24.5.1 型

型 (Ctrl + メニューキー) コマンドは JSON データ型を含むメニューを表示しています (*下のスクリーンショット*)。グリッドビュー内で現在選択されているセルに割り当てるメニューから1 つの型を選択します (メニュー キーは通常キーボードの右下で **Ctrl** キーの横に配置されています。アイコンは以下のように表示されます:)。



編集の型に関する詳細は、JSON ドキュメントコンテンツの編集 482 を参照してください。

24.5.2 子を前/後に挿入、子の付加、子の追加

The **後に挿入**、**前に挿入**、**付加**、および**この追加** コマンドはグリッドビュー内の現在の選択がコンポーネント を挿入、付加、および子として挿入されることを許可している場合有効化されます。

- **後に挿入** は同じ型のコンポーネントを選択されたコンポーネントの上のグリッドビュー内で選択されたコンポーネントとして挿入します。
- **前に挿入** は同じ型のコンポーネントをグリッド行内の選択されたコンポーネントとして選択されたコンポーネントの上に挿入します。
- **付加** 同じ型のコンポーネントをグリッド行内の選択されたコンポーネントとして選択されたコンポーネントのすべての兄弟の後に付加します。
- **この追加** は新規の子コンポーネントを最後の子として追加します。型は最後に追加された子と同じです。

JSON ドキュメント構造の編集に関する詳細はJSON ドキュメント構造の編集 481 を参照してください。

24.5.3 配列/オブジェクト内のラップ

「**配列内でラップ**」と「**オブジェクト内でラップ**」コマンドはそれぞれ選択されたコンポーネントを配列またはオブジェクト内でそれぞれラップします。JSON ドキュメント構造の編集に関する詳細はJSON ドキュメント構造の編集 481 を参照してください。

24.5.4 右から左への読み取り

このコマンドは文字列型を含むセルのために使用することができ、読み取りの向きをセルの右から開始するように切り替えます。この機能はアラブ語とヘブライ語などの言語との作業に役に立ちます。

24.5.5 移動

コンポーネントを現在の位置からグリッド内で上下左右に移動しすることが可能になり、対応するコマンドが有効化されます。移動を実行するために対応するコマンド選択してください。

24.5.6 コピー

コピー コマンドのサブメニュー内のコマンドを使用して現在の選択範囲をコピーすることができます。

コマンド	説明
JSON テキストとしてコピーする	現在の選択項目がJSON またはJSON ラインテキストとしてシリアル化されます
構成されたテキストとしてコピーする	現在の選択項目がTSV（タブ区切り値）としてシリアル化されます
イメージとしてコピーする	現在のイメージセルがイメージとしてコピーされます
XPath のコピー	XPath 3.1 構文例: **?Artists?1?Albums?1?Tracks**
JSON ポインターをコピーする	JSON ポインター構文。例: **/Artists/1/Albums/1/Tracks**

以下の点に注意してください：

- *イメージとしてコピー*はBase64 エンコード文字列をイメージファイルにエクスポートします。Base64 エンコード内で指定されているイメージ型に一致しているファイル拡張子がイメージファイルに与えられていることを確認してください。エクスポートされる文字列がBase64 エンコードではない場合、指定したファイル名を持つファイルが作成されますが、コンテンツは存在しません。
- テキストをセルに挿入するには、テキストをコピーしてセルに張り付けます。
- 例えばイメージなどをファイルからインポートするには、[編集 | 挿入 | エンコードされた外部ファイル] 768 コマンドを使用します。Base64 エンコード文字列としてイメージファイルを挿入し、この文字列をJSON グリッドビューで表示します。インポートのためにサポートされているイメージファイルフォーマットは以下の通りです: PNG、JPEG、BMP、GIF、TIFF。

24.5.7 展開、折りたたみ

フルに展開する
このコマンドは選択されているコンポーネントまたはその子孫が折りたたまれている場合に有効化されます。コンポーネントと子孫コンポーネントが展開されます。

選択されていない範囲を折りたたむ
このコマンドは選択されているコンポーネントとその祖先コンポーネント以外のすべてのコンポーネントを折りたたみます。

昇順に並べ替え、降順に並べ替え
これらのコマンドは選択されたテーブル列に従いテーブルのコンポーネントを並べ替えるために使用されます。行が隣接する場合、並べ替えはテーブル行の選択範囲にも適用することができます。

(C) 2015-2021 Altova GmbH

24.5.8 昇順に並べ替え/降順に並べ替え

コンポーネントのテーブル表示内の列が選択されている場合、これらの2つのコマンドは有効化されます。列を選択するにはヘッダーを選択します。並べ替えは列内の値をベースにしています。

24.6 DTD/スキーマメニュー

「DTD/スキーマ」メニューにはDTD やXML スキーマを使った作業を効率的に行うためのコマンドが収められています。

24.6.1 DTD の割り当て

「DTD/スキーマ | DTD の割り当て...」コマンドはXML ファイルがアクティブな時に有効になります。このコマンドによりDTD がXML ドキュメントに割り当てられ、ドキュメントの検証や、インテリジェントな編集を行うことができるようになります。コマンドを実行すると、ファイルの割り

当てダイアログが表示され、割り当てを行うDTD ファイルを指定することができます。ファイルは(参照 746 ボタンをクリックすることで) グローバルリソースやURL から、または(ウィンドウボタンをクリックすることで) XMLSpy にて既に開かれているウィンドウから選択することもできます。*…に対する相対パスにする* チェックボックスにチェックを入れることで、割り当てられたDTD ファイルに対するパスを相対パスにすることができます。操作が完了すると、割り当てられたDTD に対する参照とともに、DOCTYPE 宣言がXML ドキュメントに追加されます。DOCTYPE 宣言は以下のようになります:

```
<!DOCTYPE main SYSTEM "http://link.xmlspy.com/spyweb.dtd">
```

メモ: 新たにXML ファイルが作成されたときも、DTD を割り当てることができます。

24.6.2 スキーマの割り当て

「DTD/スキーマ | スキーマの割り当て…」コマンドはXML ファイルがアクティブな時に有効になります。このコマンドによりXML スキーマがXML ドキュメントに割り当てられ、ドキュメントの検証や、インテリジェントな編集を行うことができるようになります。コマンドを実行すると、ファイルの割り当てダイアログが表示され、割り当てを行うXML スキーマファイルを指定することができます。ファイルは(参照 746 ボタンをクリックすることで) グローバルリソースやURL から、または(ウィンドウボタンをクリックすることで)XMLSpy XMLSpy にて既に開かれているウィンドウから選択することもできます。*…に対する相対パスにする* チェックボックスにチェックを入れることで、割り当てられたXML スキーマファイルに対するパスを相対パスにすることができます。操作が完了すると、適切な名前空間とともにXML スキーマの割り当てがXML ドキュメントに追加されます。割り当ては以下のようになります:

```
xmlns="http://www.xmlspy.com/schemas/icon/orgchart"
xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
xsi:schemaLocation="http://www.xmlspy.com/schemas/icon/orgchart
    http://schema.xmlspy.com/schemas/icon/orgchart.xsd"
```

24.6.3 他の DTD を読み込み

「DTD/スキーマ | 他のDTD を取り込み…」コマンドにより、他のDocument Type Definition (DTD) や外部解析対象エンティティを、文章型定義の内部サブセットやDTD ドキュメントへインクルードします。この処理により対応する外部解析対象宣言が定義され、そのエンティティを以下のように使用することができるようになります:

```
<!ENTITY % navigation.dtd SYSTEM "S:\xml\navigation.dtd">
%navigation.dtd;
```

コマンドによりファイルの割り当てダイアログが表示され、使用しているDTD にインクルードするDTD を指定することができます。

メモ: このコマンドはグリッドビューでのみ使用することができます。

24.6.4 DTD へ移動

「DTD/スキーマ | DTD へ移動」コマンドにより、アクティブな XML ドキュメントがベースにしている DTD を開くことができます。DTD が割り当てられていない場合、エラーメッセージが表示されます。

24.6.5 スキーマへ移動

「DTD/スキーマ | スキーマへ移動」コマンドにより、アクティブな XML ドキュメントがベースにしている XML スキーマを開くことができます。XML スキーマが割り当てられていない場合、エラーメッセージが表示されます。

24.6.6 定義へ移動

「DTD/スキーマ | 定義へ移動」コマンドにより、選択されているノードに対応した、文章型定義（DTD）やスキーマ内にある要素や属性の定義が表示されます。

アイテムの定義をグリッドビューで確認するには

1. アイテムをクリックします。
2. メニューオプションの「DTD/スキーマ | 定義へ移動」を選択するか、アイコンをクリックします。

アイテムの定義をスキーマビューで確認するには

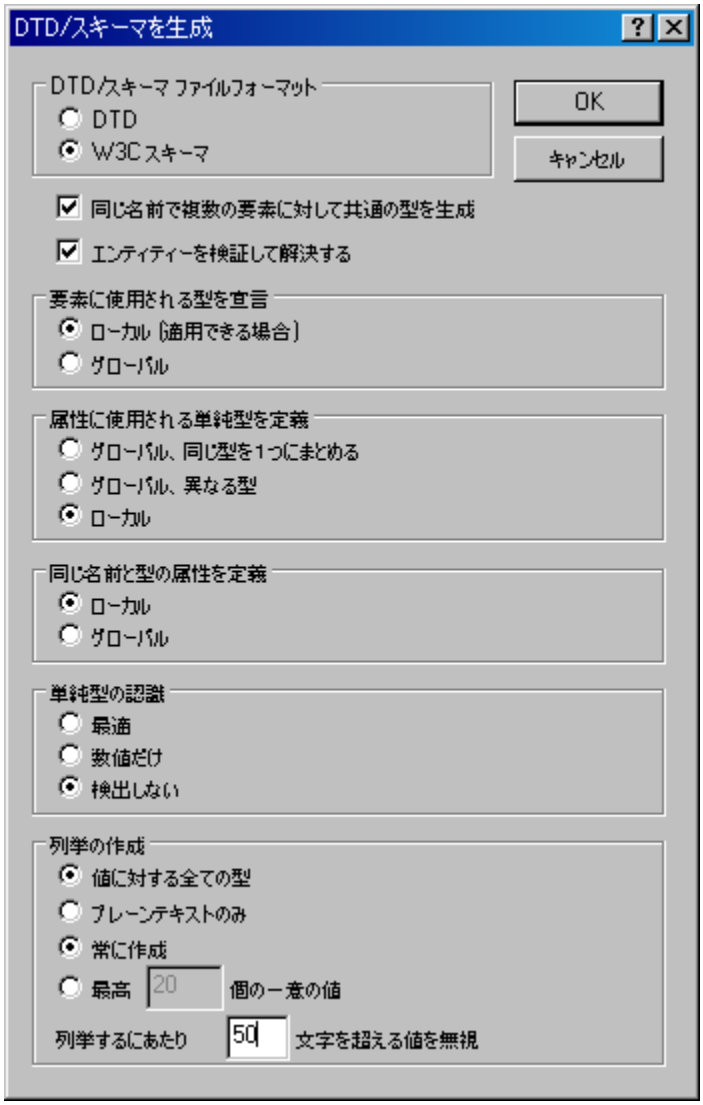
- 定義を確認したいアイテムを Ctrl キーを押下しながらダブルクリックするか、
- アイテムをクリックした後、メニューオプションから「DTD/スキーマ | 定義へ移動」を選択、またはアイコンをクリックします

どちらの場合でも、対応する DTD またはスキーマファイルが開かれ、目的の定義がハイライトされます。

24.6.7 DTD/スキーマの生成

「DTD/スキーマ | DTD/スキーマの生成」コマンドにより、XML ドキュメント（又はプロジェクトウィンドウのフォルダーに収められた複数の XML ファイル）から新たな DTD または W 3C XML スキーマを生成することができます。このコマンドは XML ドキュメントから DTD や XML スキーマを生成する際に便利な機能となります。

(C) 2015-2021 Altova GmbH



XML スキーマを生成する際には、以下のオプションが利用可能です:

- *要素* :要素の型を、ローカルかグローバルか選択することができます(要素に使用される型の定義)。同じ名前の要素が複数ある場合、これら要素に対して共通の型を宣言することができます(共通の型を生成)。
- *属性* :属性の単純型(属性に使用される単純型の定義)は、(i) 共通のグローバル型、(ii) 一意のグローバル型、または (iii) ローカル型、として定義することができます。同じ名前の属性をローカルまたはグローバルで定義することができます。
- *単純型の認識* :型の認識(単純型の認識)を、(i) できるだけ最適な認識、(ii) 数値のデータ型だけ検出、または (iii) データ型の検出無しからセットすることができます((iii) の場合は、全てのデータ型に xs:string がセットされます)。
- *エントリーの解決* :XML ドキュメントにおいて、エンティティは要素と属性の値両方にて出現しえます。そのため、それらエンティティが解決されるかどうかが、列挙の値に対して重要な意味を持ちます(エンティティを検証して解決)。更に、エンティティが解決されているかによって(特にパースされたエンティティにマークアップが含まれる場合)、コンテンツモデルが影響を受けることもあります。スキーマが生成される前に、XML ドキュメントが検証されることに注意してください。ドキュメントが妥当でない場合、スキーマの生成は行われません。
- *列挙* :値に対する全ての型、または文字列の値のみから列挙を行うことができます。

DTD を生成すると、エンティティの解決や列挙オプションが有効になります。

DTD/スキーマの生成コマンドは、通常メインウィンドウで処理されますが、アクティブなプロジェクトウィンドウ上のファイルやフォルダー、またはファイルのグループに対しても「**DTD/スキーマの生成**」コマンドを使用することができます。

1つ以上の名前空間ないで要素または属性が存在する場合、XMLSpy は個別のXML スキーマを各固有の名前空間のために生成します。このため、複数のファイルがディスク上に作成される可能性があります。

24.6.8 DTD のフラット化

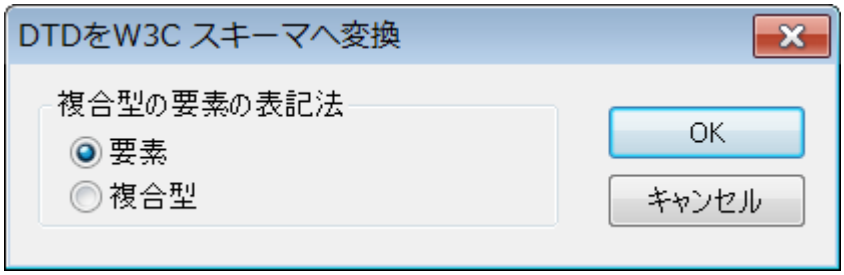
DTD のフラット化 コマンドは、DTD がアクティブなドキュメントな場合有効化されます。新規のフラットな DTD を作成し、パラメーターエンティティを削除し、モジュールのコレクションからを単一のDTD 作成します。IGNORE とマークされた箇所を抑制し、使用されていないパラメーターエンティティを削除します。

コマンドは生成されたDTD ファイルでの場所を選択する保存ダイアログをポップアップします。「**保存**」をクリックして、変換を実行します。フラット化されたDTD ファイルは生成され、XMLSpy 内で開かれます。

24.6.9 DTD をスキーマに変換

DTD がアクティブなドキュメントの場合、**DTD をスキーマに変換** コマンドが有効化されます。DTD をXML スキーマドキュメント（XSD）に変換します。

コマンドにより、複合型要素の要素 または複合型への変換を選択することのできる、DTD をW 3C スキーマに変換するダイアログがポップアップします（*下のスクリーンショット*）。「**OK**」をクリックすると、生成されたXSD ファイルを保存する場所を選択するようにプロンプトされます。変換を実行するために「**保存**」をクリックします。XMLSpy XSD ファイルが生成され、内で開かれます。



DTD をXML スキーマに変換する場合、XMLSpy により、情報が限られているため、いくつかの想定がなされます。最も顕著な例は、特定のDTD コンポーネントの値が、構文的な解析よりも、文字通り扱われる点です。これは、プログラムが複数の使用の可能性について特定できないためです。この場合、生成された変換を変更する必要があります。

拡張するために、生成された変換を注意深く確認する必要がある場合、修正の必要がある可能性のある箇所は以下にリストされています。

属性 Datatyping
DTD は10 属性 データ型のみを許可しますが、XML スキーマは、例えば、40 以上のデータ型 と派生する データ型を許可します。生成されたXML スキーマを拡張する場合、例えば、更に制約された データ型を使用する場合などがあげられます。 XML スキーマがDTD に変換される 858 場合、データ型 情報は、失われます。

名前空間
DTD は、名前空間を考慮していません。この結果、名前空間はDTD 内で指定され、要素と属性名内にハードコードされている必要があります。これは、1つのスキーマから他のスキーマに変換する際に問題を引き起こす可能性があります。

エンティティ
XML スキーマにはDTD のための一般的なエンティティ宣言に相当する宣言は存在しません。XMLSpy がDTD をXML スキーマに変換する場合、エンティティ宣言は無視されます。

解析されていないデータ宣言
DTD とXML スキーマは、解析されていないデータを扱うために異なるメカニズムを使用します。この点に関しては下で説明されています。

DTD は次のしくみを使用します:

- 表記は宣言されており、名前と識別子により構成されています。例えば、:
  `<!NOTATION gif SYSTEM "image/gif">`
- エンティティを宣言します。例えば、:
  `<!ENTITY cover_img SYSTEM "graphics/cover_img.gif" NDATA gif>`
- 通常、対応する属性上で `ENTITY` の属性の型を指定します。例えば、:`<!ELEMENT img EMPTY>`
  `<!ATTLIST img format ENTITY #REQUIRED>`

XML スキーマ内での、対応するメカニズムは、以下のとおりです:

- 表記法を宣言します。この関数は、DTD と同じ方法です。
  ```
  <xs:notation name="gif" public="image/gif"/>
  ```
  `public` 属性 は、識別子を持つためには必須であることに注意してください。任意の `system` 属性にはシステム識別子が存在し、通常、表記型のソースと作業できる実行可能です。
- `NOTATION` データ型を使用して、`notation` 宣言を与えられた属性値と関連付けます。しかしながら、`NOTATION` データ型を直接使用することはできず、`NOTATION` データ型から他のデータ型を派生する必要があります。
  ```
  <xs:simpleType name="formatType">
     <xs:restriction base="xs:NOTATION">
        <xs:enumeration value="gif"/>
        <xs:enumeration value="jpeg"/>
     </xs:restriction>
  </xs:simpleType>
  ```
- 属性を `NOTATION` データ型から派生したデータ型に関連付けます。例:
  ```
  <xs:complexType name="imgType">
     <xs:attribute name="height"/>
     <xs:attribute name="width"/>
     <xs:attribute name="location"/>
     <xs:attribute name="format" type="formatType" use="required"/>
  </xs:complexType>
  <xs:element name="img" type="imgType"/>
  ```

DTD からXML スキーマへ変換する場合、XMLSpy は以下をおこないます:

- 以下は
  ```
  <!ATTLIST image format ENTITY #REQUIRED
  ...>
  ```
  以下に変換されます
  ```
  <xs:attribute name="format" type="xs:ENTITY" use="required"/>
  ```

- 以下は
  ```
  <!NOTATION gif SYSTEM "image/gif">
  ```
  以下に変換されます
  ```
  <xs:notation name="gif" system="image/gif"/>
  ```

以下の変更を加えます:

1. `<xs:notation name="gif" system="image/gif"/>` のような表記は、`system` を `public` と置換え、任意のシステム識別子を必要な場合追加します。
2. `NOTATION` データ型から `formatType` のために説明されているとおりから派生します。
3. 派生したデータ型対応した属性と関連付けします。

メモ: XML スキーマ仕様に従い、必要に応じて外部エンティティを宣言します。

24.6.10 フラット化されたスキーマ

フラット化されたスキーマ コマンドは、XML スキーマがアクティブなドキュメントの場合有効化されます。新規のフラットな XSD を以下の方法で生成します (i) アクティブな スキーマのグローバルコンポーネントとして含まれる全てのスキーマのコンテンツを追加する、(ii) 含まれているスキーマを削除する。

コマンドは、スキーマデザインメニューのフラット化されたスキーマ 884をリダイレクトします。フラット化されたスキーマ 884 コマンドは、スキーマビューのみで使用することができるため、スキーマビューへの切り替えが問われます。 詳細に関しては、フラット化されたスキーマ 884 コマンドを参照してください。

24.6.11 スキーマを DTD に変換する

スキーマを DTD に変換する コマンドは、XML スキーマがアクティブなドキュメントな場合に有効化されています。 XML スキーマドキュメント (XSD) を DTD に変換します。

コマンドは、生成された DTD ファイルを保存する保存先を選択する保存ダイアログをポップアップします。「**保存**」をクリックして、変換を行います。DTD ファイルは生成され、XMLSpy 内で開かれます。

以下の点に注意してください:

1. XML スキーマを DTD に変換する場合、名前空間 URI または名前空間宣言内ではなく、XML スキーマ内で使用される名前空間プレフィックスは、DTD 内の対応する要素と属性の名前全体で使用されます。
2. DTD に対して XML ドキュメントの検証を行う際、名前空間は無視されます。名前空間の宣言そのものは変換されません。
3. xs:schema 要素の elementFormDefault ならびに attributeDefault 属性により、どの要素や属性が変換処理に含まれるか指定することができます。unqualified がセットされていると、変換の際にグローバルに宣言された要素や属性に対してだけプレフィックスが含まれます。qualified がセットされていると、変換の際に全ての属性や要素に対してプレフィックスが含まれます。
4. プレフィックスは、それぞれ対応する文字列とコロンに変換されます。デフォルトの名前空間にある要素や属性は、default_NS_X で始まる名前に変換されます (X には 1 から始まる数値が入り、XML スキーマで使用されているデフォルトの名前空間の数だけ数値は用意されます)。
5. DTD において、要素名はパラメーターエンティティにより構成されます。これにより、XML ドキュメント内にあるプレフィックスが変更されたときでも、DTD 内のプレフィックスを簡単に変更することができるようになります。パラメーターエンティティ定義の変更は、DTD ドキュメント内で直接行うか、XML ドキュメントの内部 DTD サブセットにてパラメーターエンティティをオーバーライドすることで行うことができます。

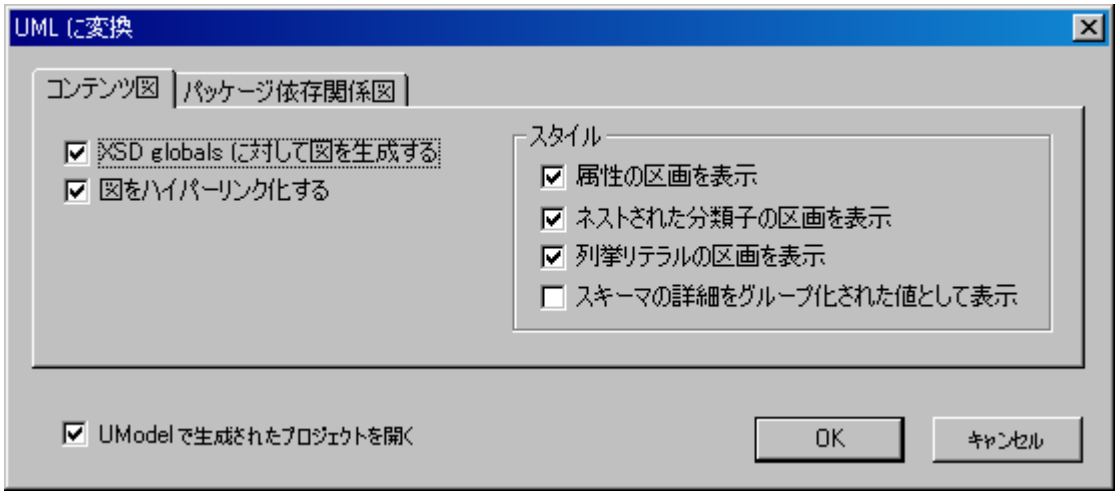
メモ: DTD における名前空間はセマンティックな意味を持っておらず、XML スキーマから変換された名前空間プレフィックスは、DTD で定義された要素名や属性名における文字列の一部でしかありません。

24.6.12 UML へ変換

「DTD/スキーマ | UML へ変換」コマンドにより、W 3C XML スキーマを Altova UModel プロジェクト (.ump) ドキュメント (UModel プロジェクト) へ変換することができます。UMP は、Altova の UML モデリングアプリケーションである Altova UModel のネイティブフォーマットです。UMP ファイルは Altova UModel にて閲覧、編集することができます。

スキーマを UML へ変換するには、以下の操作を行います:

1. スキーマが開かれた状態で、**UML へ変換**コマンドをクリックします。UML に変換ダイアログが表示されます (以下のスクリーンショットを参照)。



2. コンテンツ図タブにて、*XSD globals に対して図を生成する* オプションを選択します。このオプションにより、各グローバルコンポーネントに対してコンテンツモデル図が UModel プロジェクトにて生成されます。
3. 必要なオプションをダイアログから選択します。各オプションの説明については、以下を参照ください。
4. 作成されたプロジェクトを UModel にて直ちに確認したい場合、*UModel で生成されたプロジェクトを開く* オプションを選択してください。
5. 「OK」をクリックします。
6. 名前を付けて保存ダイアログが表示されるので、出力先フォルダーを参照し、UMP ファイルの名前を入力した後、**保存**をクリックします。

UML への変換オプション

UML に変換ダイアログでは、以下のオプションが利用できます。

コンテンツ図タブにて:

- *図をハイパーリンク化する* オプションにより、各ダイアグラムにてグローバルコンポーネントのエントリーへのリンクがモデルツリービューに作成され、スキーマ階層からコンポーネントの確認を素早く行うことができるようになります。
- *スタイル* ペインでは、区画を表示オプションにより、表示する区画を指定することができます。

パッケージ依存関係タブにて:

- *図を生成* オプションにて、パッケージ依存関係図を生成するか指定することができます。パッケージ依存関係図ではパッケージ全体の概要が表示され、パッケージコンポーネント間の関係を確認することができます。このタブにある他のオプションは、図を生成オプションが選択されていないと有効になりません。
- *ダイアグラムのパッケージをハイパーリンク化* オプションにより、パッケージ依存関係図からモデルツリービューへのリンクが作成されます。
- パッケージ依存関係図のレイアウトに対して4つのオプションが指定できます：(i) 自動レイアウトを行わない、(ii) 階層レイアウト（自動レイアウトと階層オプションを選択）、(iii) ブロック（自動レイアウトとブロックオプションを選択）、(iv) 均等に配置（自動レイアウトと有向化を強制オプションを選択）。UModel にてダイアグラムを編集することでレイアウトを修正することもできます。

メモ: UML への変換機能は、W3C XML スキーマだけをサポートしています。

24.6.13 MapForce で DB/Excel/EDI から XML を生成

「**DTD/スキーマ | MapForce で DB/Excel/EDI から XML を生成...**」コマンドにより、（アプリケーションがインストールされていれば）Altova の MapForce が起動します。MapForce ではスキーマを他の DTD や XML スキーマ、データベースへマッピングして、XML を生成することができます。

24.6.14 StyleVision で HTML/PDF/Word 出力をデザイン...

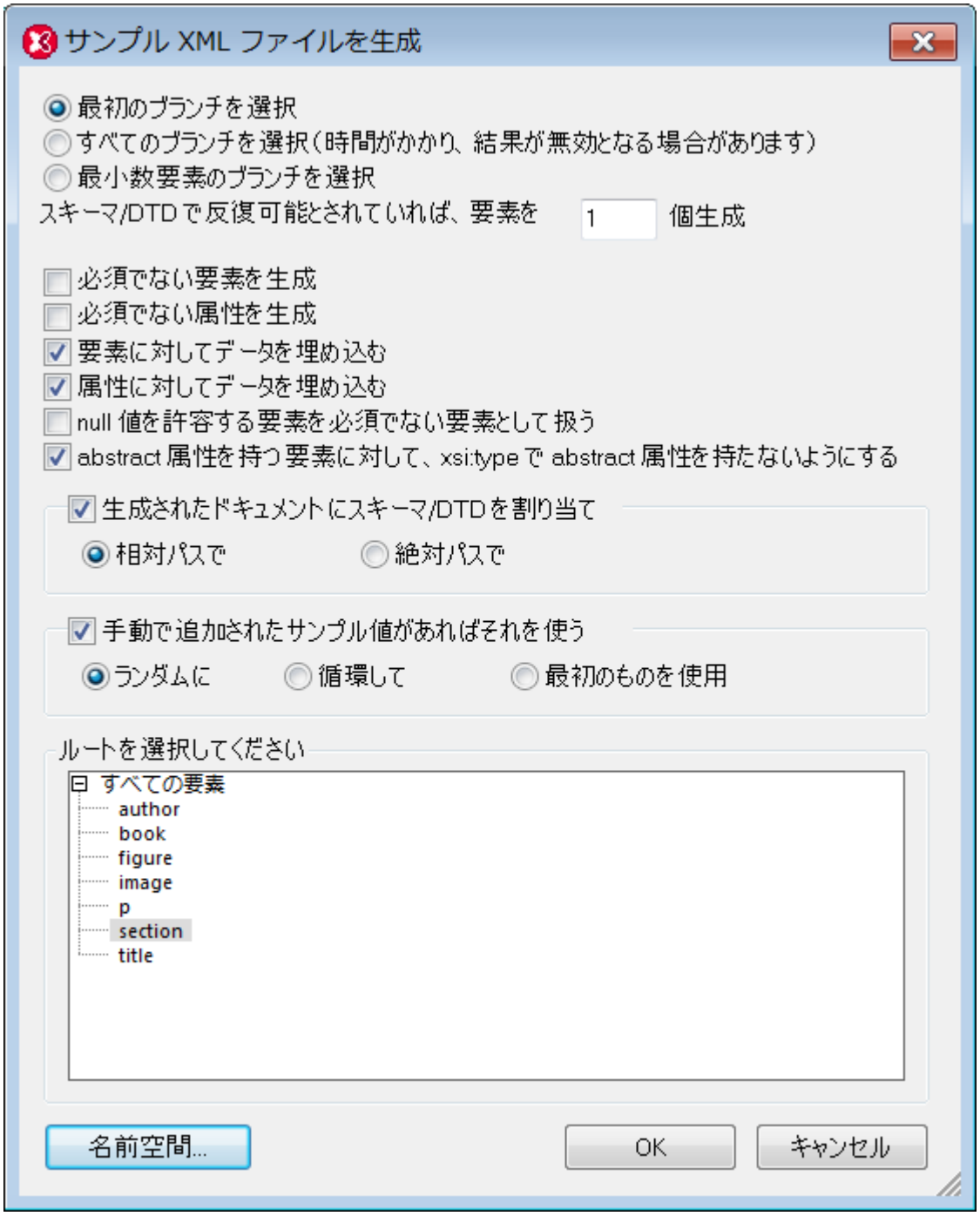
「**DTD/スキーマ | StyleVision で HTML/PDF/Word 出力をデザイン...**」コマンドにより（アプリケーションがインストールされていれば）Altova の StyleVision が起動します。StyleVision では HTML や PDF、RTF 出力に対してスタイルシートをデザインすることができます。

24.6.15 サンプル XML/JSON ファイルの生成

「**サンプル XML/JSON ファイルの生成**」コマンドがテキストビュー、グリッドビュー、とスキーマビューで有効化されており、現在アクティブなスキーマファイルをベースにして、XML または JSON インスタンスが生成されます。現在アクティブなファイルが DTD または XML スキーマの場合、XML インスタンスファイルが生成されます。現在アクティブなファイルが JSON スキーマの場合、JSON インスタンスファイルが生成されます。

サンプル XML ファイルの生成

DTD または XML スキーマがアクティブな場合、スキーマをベースにしてサンプル XML インスタンスを生成することができます。「**サンプル XML/JSON ファイルの生成**」コマンドをクリックすると、サンプルの生成のためのオプションを指定することのできるサンプル XML ファイルの生成 ダイアログが表示されます（*下のスクリーンショット*）。



選択グループの要素

選択グループは、使用される可能性のある要素のグループです。例えば、アイテムという名前の要素が以下の3 つの要素から構成される choice グループを持つと定義されている場合、(要素の maxOccurs 属性内で指定された発生に最大数である)アイテムはこれら3 つの要素を子要素として持つことができます: cd, dvd, book。

サンプルXML ファイルの生成 ダイアログ内では、以下を選択することができます (i) choice グループの最初のブランチ(要素)、(ii) 全てのブランチ、または、(iii) 子孫要素の最小数を持つブランチ。choice グループから1つのブランチが許可されているため、全てのブランチの選択は、無効なドキュメントを生成することができます。

choice グループのブランチは、(maxOccurs 値 が1より大きいため) 繰り返すことができ、ダイアログの最初のテキスト・ボックス内で、生成する繰り返すことのできる要素の数を、最大99まで、指定することができます。choice グループのmaxOccurs 属性は、unbounded として、または、大きな番号として定義されます。サンプルXML ファイルの生成 ダイアログ内で全てのブランチが選択されている場合、choice グループのmaxOccurs は、ダイアログの最初のテキスト・ボックス内で指定される繰り返すことのできる要素の数に制限されます。

必須ではない要素の生成
このオプションの有効化は、スキーマ内で定義された必須と必須ではない両方の要素を生成します。

必須ではな属性の生成
このオプションの有効化は、スキーマ内で定義された必須と必須ではない両方の要素を生成します。

繰り返しとマークされている場合 DTD 内でのX 要素の生成
このオプションの有効化により、テキスト・ボックス内に入力する繰り返すことのできる要素の数量を生成します。 choice グループ内に含まれるものも含む全ての要素に適用されます。

要素と属性にデータをフィルする
このオプションの有効化により対応する要素と属性のデータの値が挿入されます。例えば、要素がpf データ型 string として定義されている場合、要素には、string のダミーの値が与えられます。

Nillable 要素 と抽象的な型
nillable 要素のコンテンツは、必須ではないものとして扱うことができ、抽象型を持つ要素は自身のxsi:type 属性ために非抽象的な型を使用することができます。

生成された XML ファイルのためのスキーマの割り当て
XML ファイルを生成するために使用されたスキーマを相対、または、絶対パスを持つ生成されたXML ファイルに割り当てることができます。

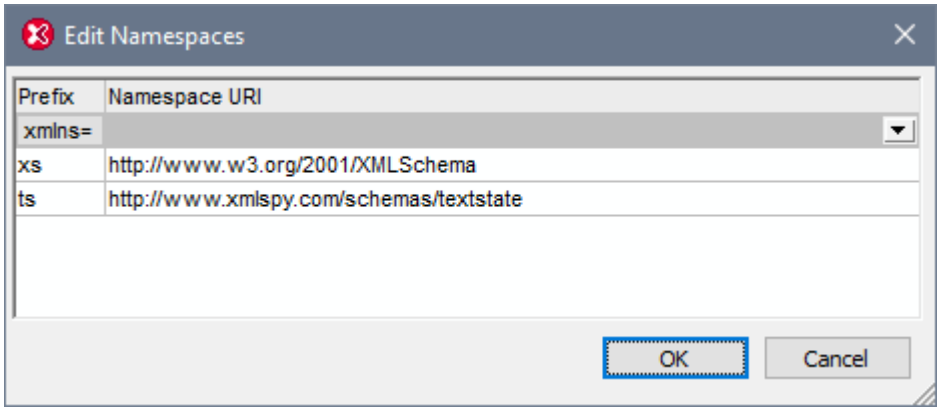
使用できる場合、手動で追加されたサンプル値を使用する
スキーマコンポーネントに割り当てられたサンプル値が存在する場合、そのコンポーネントのコンテンツまたは値として使用されます。個別のコンポーネントに関しては、サンプル値が、サンプルタブ内のファセット 入力ヘルパー 207 内で割り当てられています。使用することのできるサンプル値からの値が単一ファイルの生成のために指定することができます:

- ランダムな選択
- コンポーネントの各インスタンスのための各サンプル値。
- 最初の値

ルート要素
スキーマが1つ以上のグローバル要素を含む場合、サンプルXML ファイルのために必要なルート要素はリストから選択することができます。

名前空間
名前空間 ボタンをクリックして、名前空間の編集ダイアログを開きます (*下のスクリーンショット*)。名前空間は、スキーマ内で定義されます。また、サンプルXML ファイル内で必要とされる、標準 XML スキーマ名前空間は、このダイアログに表示されます。

 (C) 2015-2021 Altova GmbH

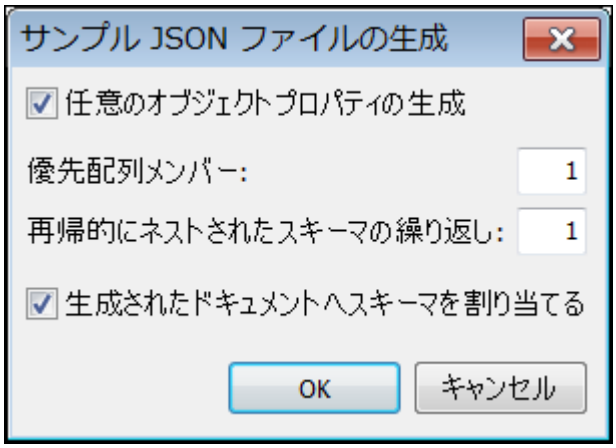


以下を編集することができます:

- ドキュメントの名前空間にバインドされている名前空間プレフィックス。このダイアログ内で設定されている名前空間プレフィックスが（生成されたXML ファイル内で使用されます。このダイアログ内で設定される名前空間プレフィックスは、対応する名前空間内のノードにプレフィックスを与えるために使用されます。例えば、スクリーンショットは、`http://www.xmlspy.com/schemas/textstate` 名前空間内のノードがサンプルファイル内で `ts:` のプレフィックスを与えられることを表示しています。
- ドキュメントの名前空間の1つを、`xmlns=` コンボボックス内でデフォルトの名前空間（`xmlns=`）を選択することにより希望する名前空間になるように設定することができます。名前空間内のノードは、デフォルトの名前空間として選択され、名前空間プレフィックスなしで生成されます。

サンプル JSON ファイルの生成

JSON スキーマがアクティブな場合、JSON スキーマをベースにした、サンプル JSON インスタンスを生成することができます。コマンドをクリックすると、サンプルの生成のオプションを指定することのできる、サンプル JSON ファイルの生成ダイアログが表示されます（*下のスクリーンショット*）



必須ではないオブジェクトプロパティ[515]、必須ではないパターンプロパティ[515]、配列の長さ[524]、および再帰的な定義の繰り返しの生成を選択することができます。アクティブな JSON スキーマが自動的に生成された JSON サンプルファイルに割り当てられる[537] かを指定する事もできます。

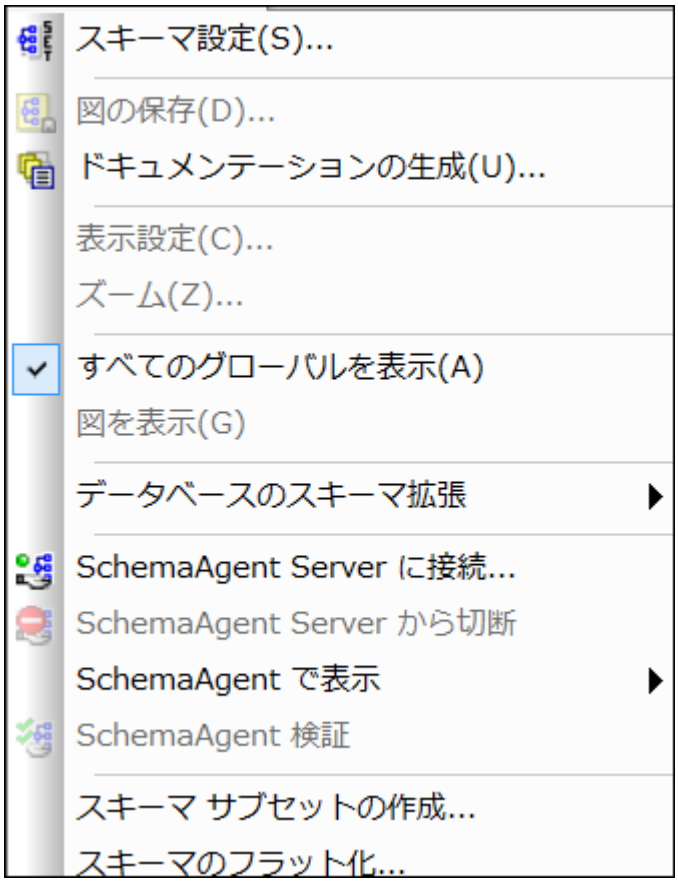
24.6.16 メモリキャッシュの破棄

「DTD/スキーマ | メモリキャッシュの破棄」コマンドにより、メモリ上にキャッシュされているスキーマ(DTD ならびにXML スキーマ)ドキュメントを破棄します。XMLSpy では検証や編集のスピードを向上させるため、最近使用されたスキーマドキュメントやが外部の解析されたエンティティをメモリに蓄えます。キャッシュされているドキュメントから得られた情報は、 定義へ移動 854 コマンドが呼び出された時などにも表示されます。

メモリにあるキャッシュを破棄することで、システム上のメモリ空き容量を増やしたり、ベースとしているスキーマの情報を更新することができます。

24.7 スキーマ設計メニュー

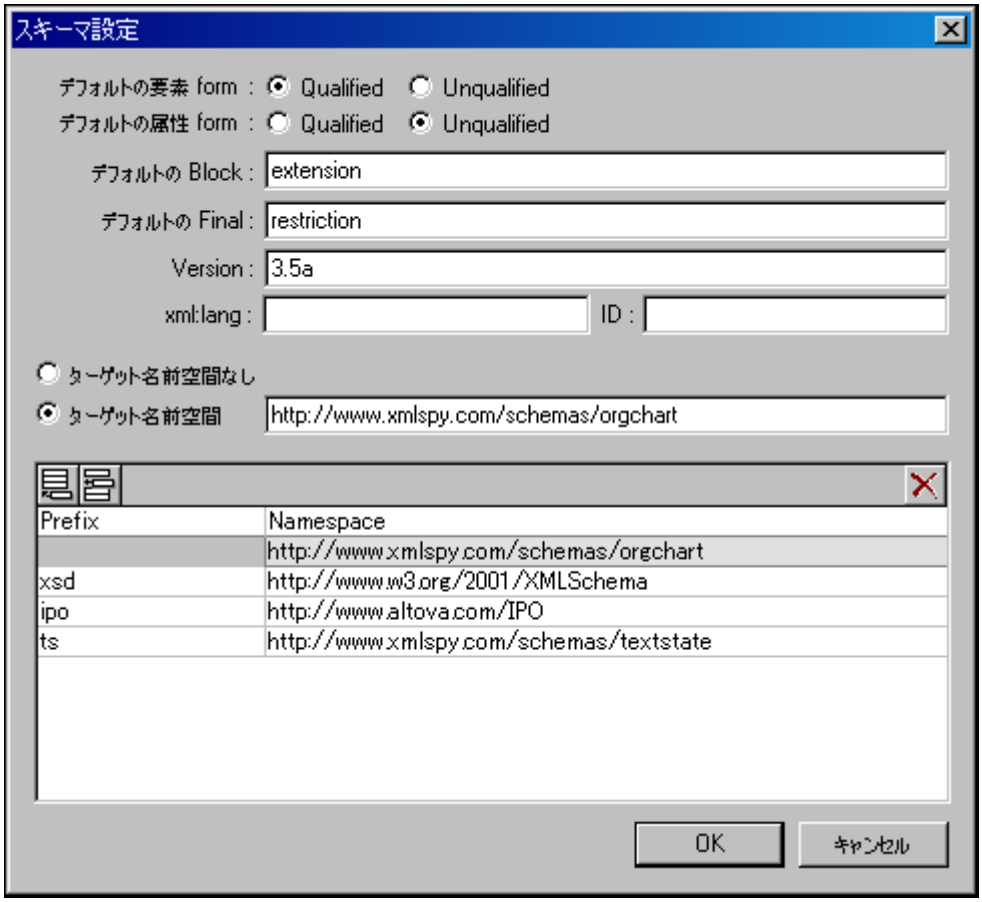
「スキーマ設計」メニューではXMLSpy のスキーマビュー設定を行ったり、XML スキーマのデザインを GUI から行うことができます。XML スキーマがスキーマビューでアクティブな状態であれば、これらメニューコマンドを使用することができます。



このセクションでは、このメニューから使用することのできるコマンドを説明します。

24.7.1 スキーマ設定

「スキーマ設計 | スキーマ設定...」コマンドを使用することで、アクティブなスキーマに対してグローバルな設定を行うことができます。これらの設定はXML スキーマドキュメント要素 xs:schema の属性やその値に対して適用されます。



スキーマ設定ダイアログ内で定義される設定（XSD モードが1.1 に設定されている場合）は次の`xs:schema` 要素を作成します。

```
<xs:schema xmlns="http://www.altova.com/schemas/org"
       xmlns:xs="http://www.w3.org/2001/XMLSchema"
       targetnamespace="http://www.altova.com/schemas/org"
       elementFormDefault="qualified"
       attributeFormDefault="unqualified"
       xpathDefaultNamespace="##targetNamespace"
       version="1.1"
       defaultAttributes="Contact">
```

メモ: スキーマ設定 ダイアログの内容は、アクティブな XSD モードにより異なります。XSD 1.0 がアクティブな モードな場合、XSD 1.1 属性は、ダイアログ内に存在しません。

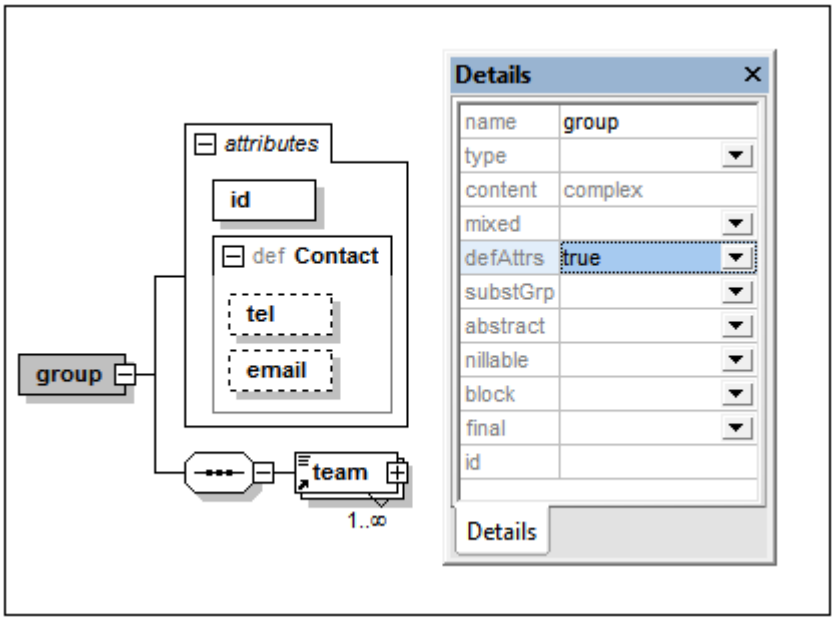
`defaultAttributes` と`xpathDefaultNamespace` 属性は、XML スキーマ1.1 の機能で、XSD 1.1 モード 153 のみで使用することができます。他の属性は、XSD 1.0 とXSD 1.1 で使用することができます。

version 属性

バージョン 属性はドキュメント バージョンです。ドキュメントのXSD バージョン 153 ではありません。

defaultAttributes 属性

defaultAttributes 属性により、隙間内の全ての複合型のデフォルトの属性グループとして属性グループを選択することができます。 デフォルトの属性 グループは、この複合型のコンテンツモデル内で表示されています。 下のスクリーンショット内では、`Contact` 属性 グループがデフォルトの属性 グループで、グループ 要素上で自動的に使用することができます(*上のスクリーンショット参照も参照してください*)。属性 グループを無効化するには、複合型 `defaultAttributesApply` を `false` に設定します。複合型の詳細入力ヘルパー内の *defAttrs* プロパティを使用してもこの設定を行うことができます(*下のスクリーンショットを参照*).



xpathDefaultNamespace 属性

`xpathDefaultNamespace` 属性は、スキーマ内で使用される XPath 式内の要素のためのデフォルトの名前空間を設定します。スキーマ設定 ダイアログ内で設定されている場合、トップレベル `xs:schema` 要素に属性が適用されます。宣言のスコープは、ドキュメント全体です。属性が許可されている箇所では、要素上の宣言が `xs:schema` 上の宣言を上書きすることが可能になります:

- `xs:assert` と `xs:assertion`
- `xs:alternative`
- `xs:selector` と `xs:` フィールド (識別制約)

上記の要素の入力ヘルパー内の XPath デフォルト名前空間を変更することができます。

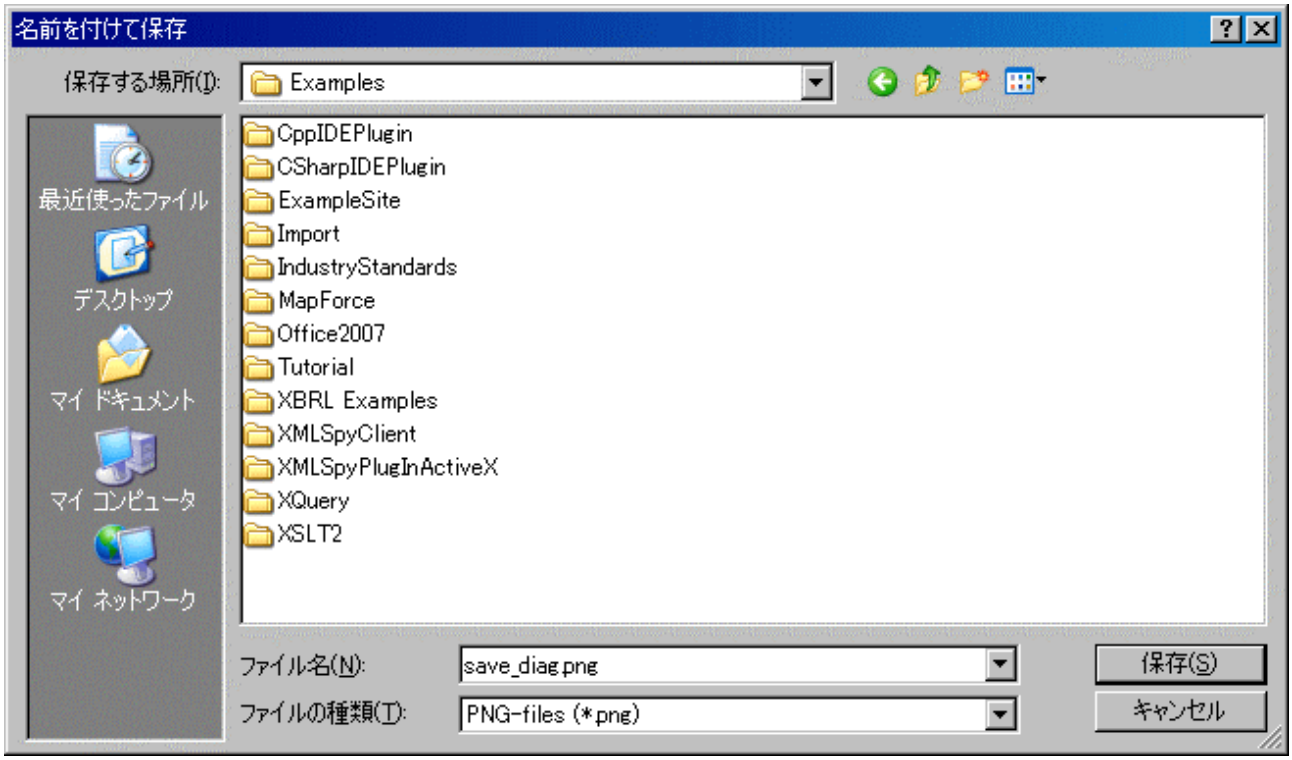
`xpathDefaultNamespace` 属性には3つの許可された値が存在します:

- `##targetNamespace`: XPath デフォルト名前空間 は、スキーマのターゲット名前空間と同じになります。
- `##defaultNamespace`: XPath デフォルト名前空間 は、スキーマのデフォルトの名前空間と同じになります。
- `##local`: XPath デフォルトの名前空間は存在しません。

XPath デフォルト名前空間がドキュメント内で宣言されていない場合、 XPath 式内のプレフィックスの無い要素には、名前空間は存在しません。 XPath デフォルト名前空間 宣言は、属性には適用されません。

24.7.2 図の保存

「スキーマ設計 | 図の保存...」コマンドにより、現在メインウィンドウに表示されているコンテンツモデルのダイアグラムを PNG フォーマットで、任意の場所に保存することができます。



24.7.3 ドキュメンテーションの生成

「スキーマ設計 | ドキュメンテーションの生成...」コマンドにより、スキーマに関する詳細なドキュメンテーションを HTML、MS Word、RTF、または PDF 形式で生成することができます(以下のスクリーンショットを参照)。このコマンドにより生成されたドキュメンテーションは、自由に使用、変更することができ、Altova から許可を得る必要はありません。(ドキュメンテーションの生成コマンドを選択したときに表示される)スキーマドキュメンテーションダイアログにて選択したコンポーネントに関するドキュメンテーションが生成されます。子要素や複合型といった関連する要素は、通常出力されたメディアのファイルにてハイパーリンク化されており、コンポーネント間のナビゲーションを行うことができるようになります。コンテンツモデルを持つコンポーネントにもコンテンツモデルの定義に対するリンクが作成されます。スキーマのドキュメンテーションはインクルードならびにインポートされたスキーマコンポーネントに対しても生成されることに注意してください。ドキュメンテーションの生成に関する様々なオプションについては、ドキュメンテーション オプション 870 にて記述されます。JSON スキーマドキュメンテーションは、JSON スキーマドキュメンテーションの生成 534 のセクションで説明されています。

ドキュメンテーションのオプション[870] は、スキーマの表示構成ダイアログ[873] 内で指定されている設定に加え適用されます

メモ: MS Word フォーマットにてドキュメンテーションの生成を行うには、お使いのコンピューターに MS Word（バージョン 2000 以降）をインストールする必要があります。

ドキュメンテーションの生成にはXMLSpy の標準デザインを使用することのできるほか、StyleVision SPS によるデザインを行うこともできます。StyleVision SPS を使用することで、生成されるドキュメンテーションのデザインや、PDF による出力を行うこともできるようになります。SPS の使用方法に関しては、ユーザー定義デザイン[872] のセクションを参照ください。

メモ: SPS を使ったスキーマドキュメンテーションの生成を行うには、お使いのコンピューターに StyleVision をインストールする必要があります。

Schema **ipo.xsd**

schema location: **C:\Program Files\Altova\XMLSPY2004\Examples\ipo.xsd**
targetNamespace: **http://www.altova.com/IPO**

Elements	Complex types	Simple types
comment	**Items**	**Sku**
purchaseOrder	**PurchaseOrderType**	

schema location: **C:\Program Files\Altova\XMLSPY2004\Examples\address.xsd**
targetNamespace: **http://www.altova.com/IPO**

Complex types	Simple types
Address	**EU-Postcode**
EU-Address	**US-State**
US-Address	

element **comment**

diagram	**comment**
namespace	http://www.altova.com/IPO
type	**string**
properties	content simple
used by	element **Items/item** complexType **PurchaseOrderType**
source	<element name="comment" type="string"/>

上のスクリーンショットは生成されたスキーマドキュメンテーションの例で、インデックス（関連するスキーマとグローバルコンポーネント）がドキュメンテーションの上部に表示されています。

メモ: ：W3C スキーマドキュメントに対するドキュメンテーションの生成には、XMLSpy アプリケーション内部に保存されているバージョンが使用されます。その結果、外部で新たに編集されたスキーマに関する情報は、ドキュメンテーションに反映されません。

24.7.3.1 ドキュメンテーションオプション

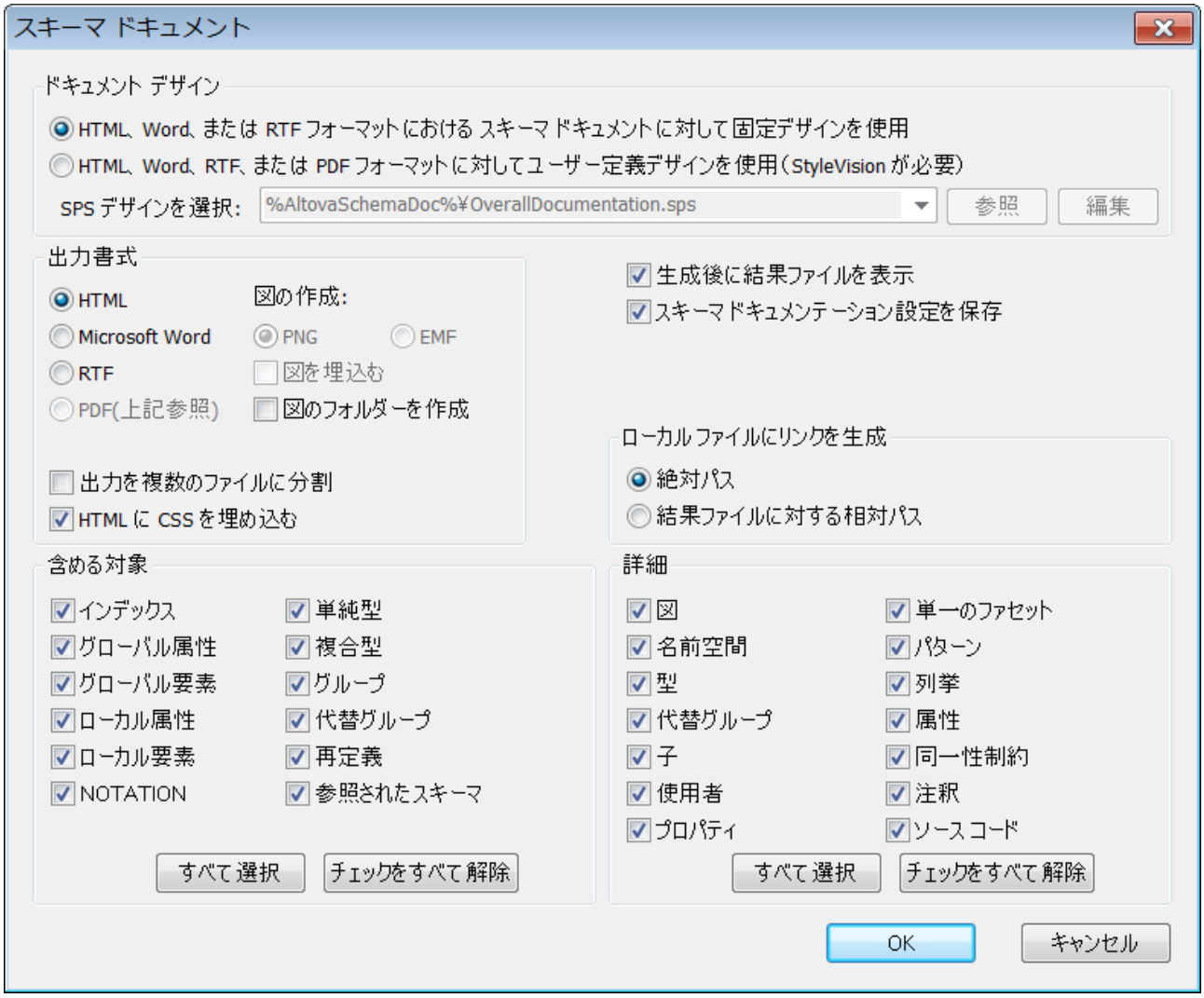
スキーマデザイン | ドキュメンテーションの生成 コマンドは、アクティブなスキーマの詳細なドキュメントを生成します: XML スキーマまたは JSON スキーマ。このセクションでは、XML スキーマドキュメンテーションの生成について説明されています。 スキーマドキュメンテーションの生成の過程に類似しています。JSON スキーマドキュメンテーションの生成と、ドキュメント生成設定の詳細に関しては、以下のセクションを参照してください: JSON スキーマドキュメンテーションの生成 534.

スキーマドキュメンテーションの生成

XML スキーマドキュメントがアクティブで、「**ドキュメンテーションの生成**」コマンドを選択すると、スキーマドキュメンテーションダイアログが表示され、ドキュメンテーションに関するオプションを選択することができます(以下のスクリーンショットを参照)。

ダイアログのドキュメンテーションデザインペインでは、ドキュメンテーションの生成に XMLSpy の固定デザインを使用するか、StyleVision SPS により作成されたカスタムデザインを使用するか選択することができます。目的のオプションを選択してください。PDF の出力は、StyleVision SPS を使ってドキュメンテーションの生成でしか有効にならないことに注意してください。ユーザー定義されたデザインの使用方法については、ユーザーにより定義されたデザイン 872 のセクションを参照ください。

(C) 2015-2021 Altova GmbH



以下に*スキーマドキュメンテーションダイアログ*にて設定することにできるオプションについて記します:

- 出力書式ペインにて、HTML、Microsoft Word、RTF、または PDF から目的のフォーマットを指定します（PDF 出力書式は StyleVision SPS を使ったドキュメンテーションの生成でしか使用できません）。OK をクリックすると、出力ファイルの名前と場所を入力するよう求められます。
- 固定デザインを使用した場合、.doc ファイル拡張子の Microsoft Word ドキュメントが生成され、StyleVision SPS を使った生成を行うと、.docx ファイル拡張子のドキュメントが生成されます。
- ドキュメンテーションは単一のファイルとして生成することも、複数のファイルに分割して生成することもできます。複数のファイルが生成される場合、各ファイルが個々のコンポーネントに対応します。出力ファイルに含められるコンポーネントは、*含める対象* ペインにて指定することができます。固定デザインを使用する場合、ドキュメント間には自動的にリンクが作成されます。
- HTML 出力では、CSS のスタイル定義を、外部 CSS ファイル内に保存するか、HTML ファイル内（<head> 要素の内部）に保存するか選択することができます。外部 CSS ファイルを作成する場合、.css という拡張子を持った HTML ファイルと同名のファイルが作成されます。*HTML に CSS を埋めこむ* オプションにより、オプションの指定を行うことができます。
- *図を埋めこむ*オプションはMS W ord、RTF、PDF 出力に対して有効になります。このオプションが有効になっていると、ダイアグラムが PNG またはEMP 形式でファイルの中に埋めこまれます。このオプションが指定されていない場合、PNG ならびにEMP ファイルが作成され、オブジェクトリンクを使ってファイルの中に表示されます。

- HTML 形式の出力が指定された場合、全てのダイアグラムは外部の PNG ファイルとして作成されます。*図のフォルダーを作成* チェックボックスにチェックが入っている場合、HTML ファイルと同じ場所にフォルダーが作成され、PNG ファイルがそのフォルダー内部に保存されます。このフォルダーの名前は、`HTMLFilename_diagrams` になります。*図のフォルダーを作成* チェックボックスにチェックが入っていない場合、PNG ファイルは HTML ファイルと同じ場所(フォルダー)に保存されます。
- ダイアグラムのイメージや外部 CSS ファイルといったローカルファイルへのリンクは、相対パスか絶対パスか指定することができます。*ローカルファイルにリンクを作成* ペインにて、適切なラジオボタンを選択してください。
- *含める対象* ペインでは、ドキュメンテーションに含めるアイテムを選択します。インデックスオプションにより、関係する全てのスキーマが、グローバルコンポーネントとともにファイル上部に表示されます。**全てチェック**と**全てクリア**ボタンにより、ペインにある全てのオプションを選択または選択解除することができます。含むオプションは、視覚的な定義内のアイテムの表示に影響を与えません。 スキーマ表示の構成 873 ダイアログにより視覚的な定義内のアイテムは管理されています。スキーマアイテムのグラフィカルな表示内の属性の表示を無効化する場合、スキーマ表示の構成 873 ダイアログ内の属性オプションのチェックを解除してください。
- *詳細* ペインでは、各コンポーネントに含める詳細情報を指定します。ドキュメンテーションに含める項目を選択してください。**全てチェック**と**全てクリア**ボタンにより、ペインにある全てのオプションを選択または選択解除することがで
- *生成後にファイルを表示* オプションは全ての出力書式に対して使用することができます。このオプションにチェックが入っていると、生成されたファイルがブラウザビュー(HTML 出力)、MS Word(MS Word 出力)、.rtf ファイルや .pdf ファイルの(それぞれ RTF 出力と PDF 出力)デフォルトアプリケーションにて表示されます。ドキュメンテーション内のリンクを相対パスにするか、絶対パスにするか選択することもできます。

パラメーター値

StyleVision SPS にて以上のパラメーター定義が含まれる場合、**OK** をクリックした際に SPS にて定義された全てのパラメーターがダイアログボックスにて表示されます。このダイアログにてパラメーター値を入力すると、SPS に割り当てられたデフォルトのパラメーター値をオーバーライドすることができます。

24.7.3.2 ユーザーにより定義されたデザイン

標準で用意されている XMLSpy の固定デザインの代わりに、カスタマイズしたスキーマドキュメンテーションのデザインを作成することができます。カスタムデザインは、出力ドキュメントのためのデザインテンプレートである StyleVision SPS 内で作成することができます。

SPS の作成

StyleVision Power Stylesheet (SPS) は、Altova の StyleVision により作成することができます。スキーマドキュメンテーションを生成するための SPS は、スキーマドキュメンテーションの構造を指定した XML スキーマをベースにする必要があります。このスキーマは SchemaDocumentation.xsd という名前で、XMLSpy のパッケージに同封されています。ファイルは以下のフォルダー以下に収められています:`C:\Documents and Settings\<username>\My Documents\Altova\XMLSpy2021\Documentation\Schema`.

StyleVision にて SPS を作成する場合、`SchemaDocumentation.xsd` スキーマから得られたノードがデザインテンプレートに表示され、スタイルとプロパティが割り当てられます。リンクやテーブル、イメージといった追加コンポーネントも SPS デザインに追加することができます。このようにして、出力ドキュメント全体を SPS にてデザインすることができます。StyleVision における SPS デザインの作成方法に関する詳細は、StyleVision のユーザーマニュアルを参照ください。

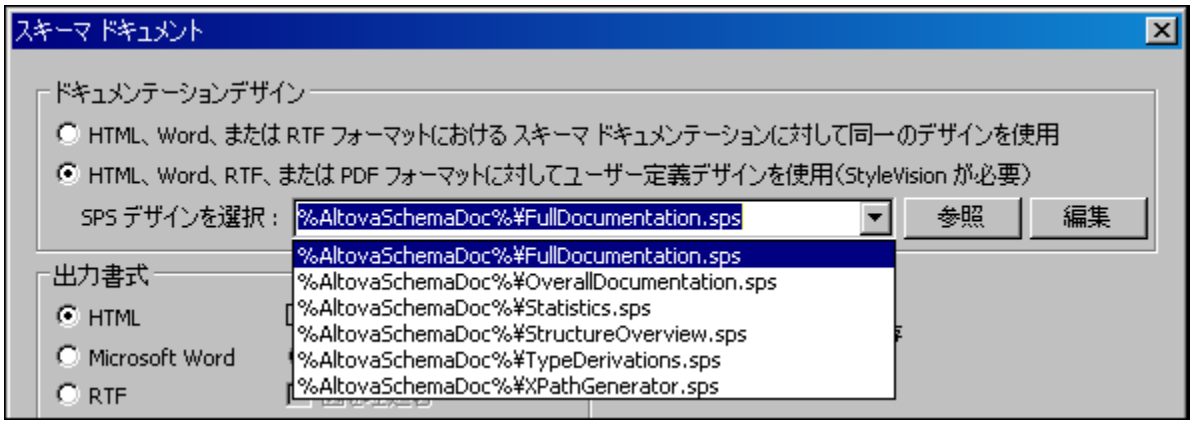
SPS によるスキーマドキュメンテーションの生成により、ドキュメンテーションのデザインを完全にコントロールすることができます。ユーザー定義の SPS を使用することで、PDF によるスキーマドキュメンテーションの生成を行うこともできます。XMLSpy の固定デザインを使用する場合、PDF 出力オプションを選択することはできません。

スキーマドキュメンテーションに使用する SPS の指定

SPS が作成されると、スキーマドキュメンテーションの生成に使用することができます。スキーマドキュメンテーションの生成に使用する SPS は、(「**スキーマ設計 | ドキュメンテーションの生成**」コマンドからアクセスすることができる)スキーマドキュメンテーションダイアログにて選択することができます。ダイアログのドキュメンテーションデザインペインにて、*ユーザー定義デザインを使用* ラジオボタンを選択します(*以下のスクリー*

ンショットを参照）。参照ボタンをクリックすることで、目的のSPS ファイルを選択することができます。ダイアログの「OK」ボタンをクリックして、表示される保存ダイアログにて出力ファイルの保存先フォルダーを選択、ファイル名を入力します。

メモ　SPS ファイルでは、ベースとなっている次のスキーマを正しく指定する必要があります: `SchemaDocumentation.xsd` (*上記参照*)。



XMLSpyを使用して次の編集可能な SPS デザインの生成を行うことができます `Altova\XMLSpy2021\Documentation\Schema\` subfolder of the マイドキュメントフォルダー[27]内で見つけることができます:

- `OverallDocumentation.sps`：スキーマに関する完全なドキュメンテーションを生成します。
- `Statistics.sps` ：グローバルやローカル要素、属性、属性グループ、単純型、複合型を、メインスキーマと関連する各スキーマごとに表示します。
- `StructureOverview.sps` は構成できる深さまでグローバル要素の構造と複合型の構造をを生成します。
- `TypeDerivations.sps` は単純型および複合型をツリーの書式で直接的にまたは間接的に生成しリストします。
- `XPathGenerator.sps`, は構成可能な深さまで可能なすべてのXPath ステートメントを生成します。

これらのファイルは他のSPS ファイルと共に *ユーザー定義を使用* オプションのコンボボックス内で使用できるようになります（*上のスクリーンショットを参照*）。

ドキュメントデザインパネル内の「編集」ボタンのクリックするとStyleVision は選択されたSPS をStyleVision ウィンドウ内で開きます。StyleVision 内で結果ドキュメントをプレビューするには、作業 XML ファイルが必要になります。上にリストされるSPS デザインは既に `Sample.xml` という名前のサンプルXML ファイルに割り当てられています。このファイルはマイドキュメントフォルダー[27]，内の以下のサブフォルダー内で見つけることができます:

```
Altova\XMLSpy2021\Documentation\Schema\SampleData
```

メモ　SPS によるスキーマドキュメンテーションの生成には、お使いのコンピューターにStyleVision をインストールする必要があります。

24.7.4　表示設定

「スキーマデザイン | ビューの構成」コマンドは、コンテンツモデルビューで有効化されており、コンテンツモデルビューの構成を行うことができます。コマンドをクリックすると、XMLSpy ウィンドウの右下のスキーマ表示の構成 ダイアログが開かれ、ダイアログ内に設定が入力されると、その効果を確認することができます。ダイアログの「OK」ボタンをクリックする、ダイアログ設定が有効になり、XML スキーマファイルのコンテンツモデルビューに対して適用されます。これらの設定は、スキーマドキュメンテーション出力[868]と印刷出力にも適用されます。例えば、スキーマドキュメンテーション出力内のスキーマアイテムのグラフィカルな表示内の属性の表示を無効化する場合、スキーマ表示の構成[873]ダイアログ内の属性オプションのチェックを解除してください（*下のスクリーンショット*）。

メモ: JSON の構成の方法に関しては、以下のセクションを参照してください: JSON スキーマデザインビューの構成 532。

コンテンツモデルのためにプロパティ記述子ラインを定義する

コンテンツモデルビューに表示される要素ならびに属性のプロパティを定義することができます。これらプロパティは、コンポーネントボックスにてグリッド線とともに表示されます。

プロパティ記述子ラインを定義するには、以下を行います:

1. 「**スキーマデザイン | ビューの構成**」を選択します。スキーマ表示の構成 ダイアログが表示されます。
2. 「**要素**」または「**属性**」タブ内で、追加 または 挿入 アイコンをクリックし、プロパティ記述子ラインを追加します。ラインがダイアログ、およびコンテンツモデルビュー内の要素ボックスに追加されます。
3. コンボボックスから、表示するプロパティを選択します。*スクリーンショットを参照してください*。
4. ステップ 2 と3 を必要に応じて繰り返してください。

コンテンツモデルビューが更新され、定義されたプロパティ記述行が、要素に対して表示されます。

メモ: 属性のプロパティは、ダイアグラムに属性が表示されている時だけ表示されます(コンテンツモデルビュー以下にあるペインに属性が表示されている場合、意味を持ちません) ビューの構成が定義された後、開かれている全てのコンテンツモデルビューに対して設定が適用されます。

 (C) 2015-2021 Altova GmbH

コンテンツモデルビューからプロパティ記述子ラインを削除する

個々のプロパティ記述行を削除するには、スキーマ表示構成ダイアログにて削除するプロパティ記述行を選択し、削除アイコン をクリックします。

コンテンツモデルビューの構成のための設定

コンテンツモデルビューの設定は、スキーマ表示構成ダイアログにて行うことができます。プロパティ記述行の表示については上記を参照ください。以下はその他の設定に関する記述です。

単一行の設定
プロパティの記述に単一もしくは複数のコンテンツを表示するのか、個々の行が全ての要素に表示されるのか（それともプロパティが含まれる場合だけ表示されるのか）指定することができます。適切なラジオボタンを使用することで、設定の定義を行うことができます。これら2つの設定は、各行に対して個別に指定することができます（目的の行を選択して、設定を行ってください）。

共有行の設定
このオプションでは、行の説明を表示するか指定することができます。

幅
スライダーにより、コンテンツモデルビューに表示される要素長方形の最小ならびに最大サイズを指定することができます。記述が全て表示されないときや、表示を標準化するのにこの値を変更してください。

距離
スライダーにより、画面上に表示されている各要素間の距離を指定することができます。

図に表示
説明チェックボックスでは、説明テキストの表示を指定したり、その幅をスライダーにより指定することができます。代替グループチェインの表示を指定することもできます。属性ならびに同一性制約のチェックボックスにチェックが入っていれば、それらのボックスがコンテンツモデルビューにそれぞれ表示され、チェックが入っていない場合は、コンテンツモデルビューの下部にあるペインのタブとして表示されます。

描写方向'
画面上に表示される属性ツリーの描画方向を、水平または垂直方向に指定することができます。

ダイアグラムにてコンテンツモデルの編集E
要素のプロパティは、コンテンツモデルダイアグラムにて直接変更することができます。編集したいプロパティをダブルクリックして、データを入力してください。選択が有効になっている場合、ドロップダウンリストが表示され、オプションの選択を行うことができます。それ以外の場合は、値を入力して「Enter」にて確定します。

スキーマ表示の構成 ダイアログ内のボタン
ダイアログには以下のボタンが用意されています:

- 開く/保存ボタンにて、ここでなされた設定のロードや保存を行うことができます。
- 定義済みボタンにて、表示構成をデフォルト値に戻します。
- 全てクリアボタンにより、全エントリーのリストを空にします。

スマート制約の有効化

スマート制約 215 を有効にするには、スマート制約の有効チェックボックスにチェックを入れてください。

24.7.5 ズーム

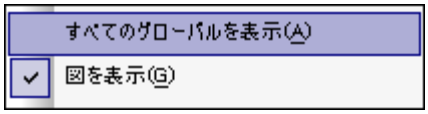
「スキーマ設計 | ズーム」コマンドにより、コンテンツモデルビューの表示倍率を変更することができます。巨大なコンテンツモデルの全体をズームアウトして、メインウィンドウに表示するような場合に、この機能を使用することができます。ズームは、実際の大きさの10% から200% の間で行うことができます。

(C) 2015-2021 Altova GmbH

ズームインやズームアウトを行うには、スライダーをドラッグするか、入力ボックスをクリックして、パーセンテージの値を入力してください。

24.7.6　全てのグローバルを表示

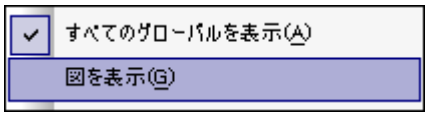
「**スキーマ設計 | 全てのグローバルを表示**」コマンドにより、コンテンツモデルビュー[168] から スキーマ概要[156] へ表示を切り替え、スキーマ内にある全てのグローバルコンポーネントを表示することができます。このコマンドは、図を表示コマンドと対になっており、現在選択されている表示方法が、チェックマークにより示されます（スクリーンショットを参照）。



その他にも、スキーマ概要の上部にある**全てのグローバルを表示**アイコン を使用して、スキーマ概要への切り替えを行うことができます。

24.7.7　図を表示

「**スキーマ設計 | 図を表示**」コマンドにより、（選択されたコンポーネントにコンテンツモデルがある場合）選択されたグローバルコンポーネントの コンテンツモデルビュー[168] への切り替えを行います。コンテンツモデルが含まれるグローバルコンポーネント（複合型、要素、要素グループ）は、 アイコンが左側に表示されます。このコマンドは、全てのグローバルを表示コマンドと対になっており、現在選択されている表示方法が、チェックマークにより示されます（スクリーンショットを参照）。



その他にも、以下の方法を使用することでコンテンツモデルビューへの切り替えを行うことができます:

- コンテンツモデルを表示したいコンポーネントの隣にある アイコンをクリック
- （画面右上にある）コンポーネントナビゲーション入力ヘルパーに表示されているコンポーネント名をダブルクリック

24.7.8　データベースに対するスキーマ拡張

このメニューコマンドでは、Oracle ならびにMS SQL Server スキーマ拡張に対するコマンドが収められています。

- Oracle スキーマ拡張を有効にする[878]

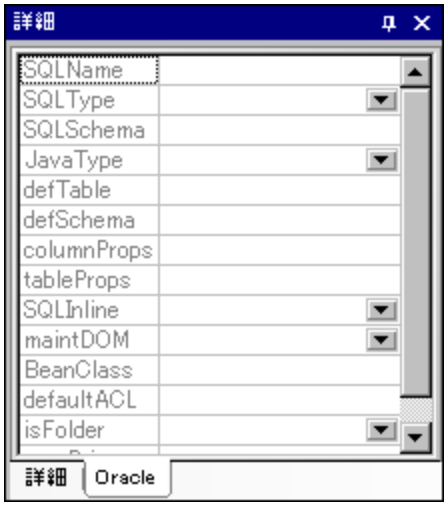
- Oracle スキーマの設定 879
- Microsoft SQL Server スキーマ拡張を有効にする 879
- 名前付きスキーマリレーションシップ 880
- 名前なし要素のリレーションシップ 880

24.7.8.1 Oracle スキーマ拡張を有効にする

XMLSpy では、Oracle 9i プロジェクト XDB にて使用することのできる Oracle スキーマの拡張がサポートされます。これらスキーマの拡張を使用することで、Oracle 9i プロジェクト XDB にて XML ドキュメントが保管される方法を設定することができます。これら XML ドキュメントへは SQL の問い合わせやそれまで使用されてきたツールによりアクセスすることができます。詳細については Oracle ウェブサイト を参照ください。

「**スキーマ設計 | データベースのスキーマ拡張 | Oracle スキーマ拡張を有効にする**」コマンドを選択すると、以下のことが起こります:

- XDB 名前空間が schema 要素に対して宣言されます: xmlns:xdb="http://xmlns.oracle.com/xdb"
- Oracle タブが詳細入力ヘルパーに表示され、属性(XDB 固有の属性も含む)を xsd:complexType や xsd:element といったスキーマ要素に加えることができるようになります。

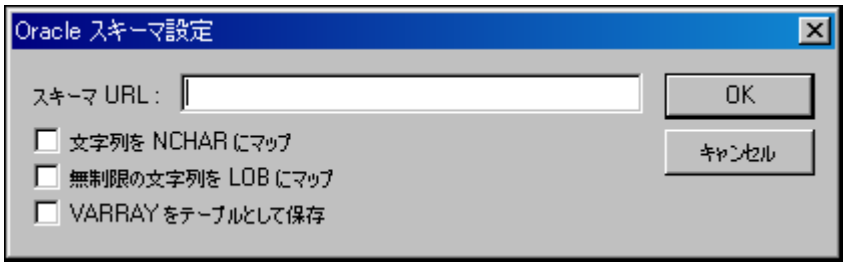


Oracle の拡張は、複合型、要素、そして属性に対して定義することができます。入力ヘルパーは、XMLSpy にあるその他の入力ヘルパーと同じように使用することができます。

メモ: このメニューコマンドはオンやオフに切り替えることが可能で、拡張を有効または無効にすることができます。Oracle 拡張が有効になっている場合、コマンドの左にチェックマークが表示されます。(有効になっているコマンドをクリックすることで)Oracle 拡張を無効にすると、XDB 名前空間ならびにファイル内にある全ての XDB 拡張が削除されます。この操作はやり直すことができないため、警告メッセージが表示されます。

24.7.8.2 Oracle スキーマの設定

「スキーマ設計 | Oracle スキーマの設定」コマンドにより、Oracle スキーマ拡張に対してグローバルな設定を定義することができます。



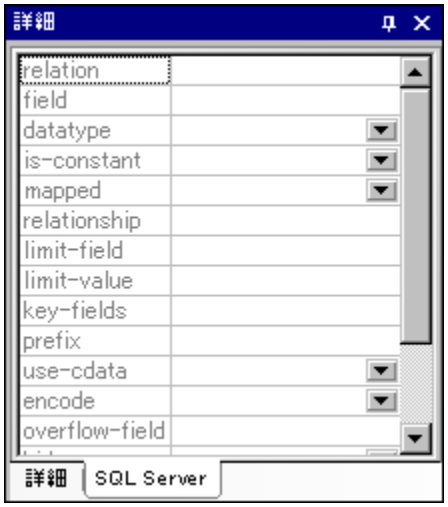
このダイアログへアクセスするには、(Oracle スキーマ拡張を有効にする 878 コマンドにより) Oracle スキーマ拡張が有効になっている必要があります。

24.7.8.3 Microsoft SQL Server スキーマ拡張を有効にする

XMLSpy では Microsoft SQL Server にて使用することができる Microsoft SQL Server 2000 スキーマ拡張がサポートされます。これらスキーマ拡張を使用することで、Microsoft SQL Server にて XML ドキュメントが保管される方法を設定、カスタマイズすることができます。これら XML ドキュメントへは SQL の問い合わせや、それまで使用されてきたツールによりアクセスすることができます。詳細については Microsoft ウェブサイト を参照ください。

「スキーマ設計 | Microsoft SQL Server スキーマ拡張を有効にする」コマンドを選択すると、以下のことが起こります:

- SQL Server 名前空間が schema 要素に対して宣言されます: xmlns:sql="urn:schemas-microsoft-com:mapping-schema"
- SQL Server タブが詳細入力ヘルパーに表示され、属性や xsd:element といったスキーマ要素を追加することができるようになります。



SQL Server 拡張は、スキーマコンポーネントに対して定義することができ、コンポーネントが選択された時、詳細入力ヘルパーに表示されます。入力ヘルパーは、XMLSpy にあるその他の入力ヘルパーと同じように使用することができます。

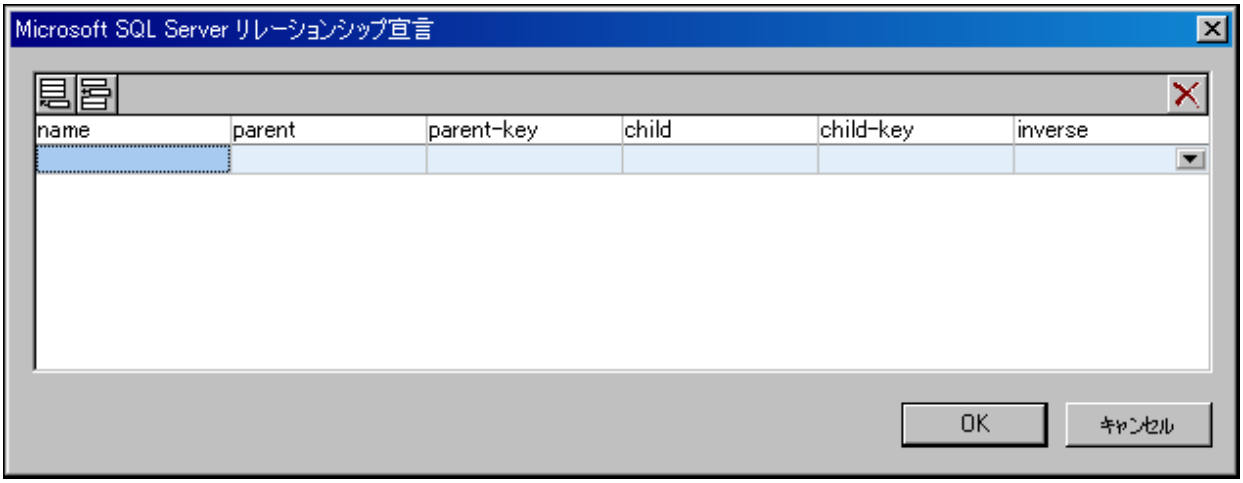
メモ: このメニューコマンドはオンまたはオフに切り替えることが可能で、拡張を有効または無効にすることができます。SQL Server 拡張が有効になっている場合、コマンドの右にチェックマークが表示されます。(有効になっているコマンドをクリックすることで)SQL Server 拡張を無効にすると、SQL Server 名前空間宣言ならびにファイル内にある全てのSQL 拡張が削除されます。この操作はやり直すことができないため、警告メッセージが表示されます。

24.7.8.4 名前付きスキーマリレーションシップ

「**スキーマ設計 | 名前付きスキーマリレーションシップ**」コマンドにより、名前付きリレーションシップの定義から、ドキュメント階層の作成に必要な情報を得ることができるようになります。このメニューオプションへアクセスするには、「**SQL Server スキーマ拡張を有効にする**」メニューオプションによりSQL Server スキーマ拡張を前もって有効にしておかなければなりません。

名前なしスキーマリレーションシップを作成するには:

1. 挿入 または追加アイコン をクリックして、新たな行をダイアログボックスへ追加します。
2. フィールドをクリックして、対応するリレーションシップ名を入力します。
3. 「OK」をクリックして確認します。



この操作により、SQL リレーションシップ要素が作成され、名前空間宣言のすぐ後に配置されます。

メモ: 削除アイコン をクリックすることで、ダイアログボックスから行を削除することができます。

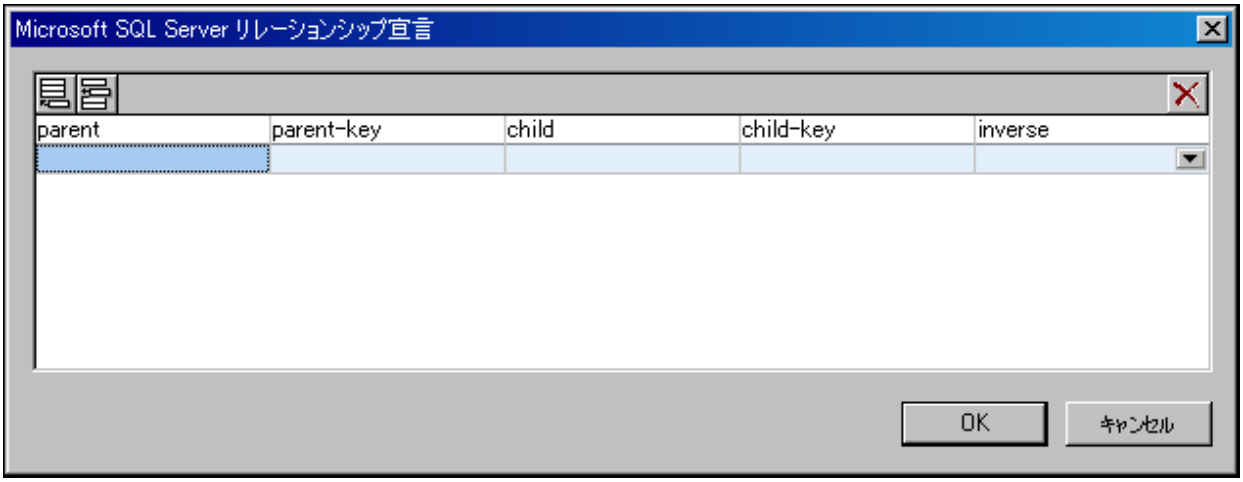
24.7.8.5 名前なし要素のリレーションシップ

名前なし要素リレーションシップコマンドにより、名前無しリレーションシップの定義から、ドキュメント階層の作成に必要な情報を得ることができるようになります。このメニューオプションへアクセスするには、「**SQL Server スキーマ拡張を有効にする**」メニューオプションによりSQL Server スキーマ拡張を前もって有効にしておかなければなりません。

名前なしスキーマリレーションシップを作成するには:

1. 挿入 または追加アイコン をクリックして、新たな行をダイアログボックスへ追加します。
2. フィールドをクリックして対応する名前を入力します。

3. 「OK」をクリックして確認します。

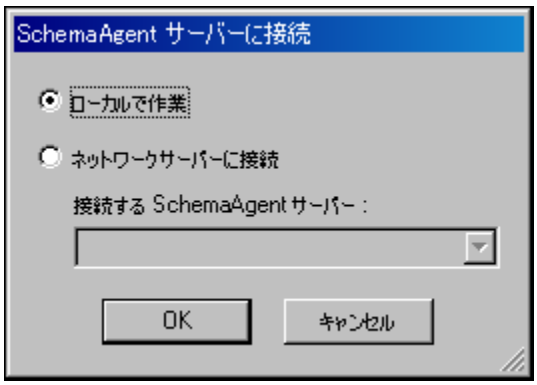


この操作により、SQL リレーションシップ要素が作成され、名前空間宣言のすぐ後に配置されます。

メモ: 削除アイコン をクリックすることで、ダイアログボックスから行を削除することができます。

24.7.9 SchemaAgent Server に接続

「スキーマ設計 | **SchemaAgent Server** に接続」コマンドは、XML スキーマドキュメントがアクティブなときに使用することができ、このコマンドにより SchemaAgent Server へ接続することができます。SchemaAgent Server への接続を行うには、お使いのコンピューターに SchemaAgent がインストールされていなければなりません。このコマンドをクリックすると、SchemaAgent サーバーへ接続ダイアログが表示されます(以下のスクリーンショットを参照):



ローカルのサーバー(Altova SchemaAgentに同封されている SchemaAgent サーバー)または**ネットワークサーバー**(無料で使用することができる Altova SchemaAgent Server 製品)を使用することができます。「**ローカルで作業**」を選択して「OK」をクリックすると、ローカルサーバーの SchemaAgent が起動され、接続が確立されます。「**ネットワークサーバーに接続**」を選択した場合、接続を確立するための SchemaAgent Server を前もって起動しておく必要があります。

SchemaAgent Server へ接続した場合、XMLSpy はSchemaAgentt クライアントとして動作して、パワフルなスキーマ編集ならびに管理機能が提供されます。SchemaAgent やSchemaAgent Server のインストール、そしてSchemaAgent Server への接続方法に関する詳細については、このユーザーマニュアルのDTD/XML スキーマセクション以下にある SchemaAgent 289 を参照ください。これら2つの製品のインストールや動作に関する更に詳細な情報については、製品に付属するSchemaAgent ユーザーマニュアルを参照ください。

SchemaAgent Server へ接続すると、メインウィンドウ上部のバーに、接続に関するメッセージが表示されます。これで、XMLSpy が接続しているSchemaAgent サーバーにて定義された検索パスやフォルダー内にある全てのスキーマやスキーマコンポーネントへのアクセスが行えるようになります。

注意：接続を行うには、XMLSpy がインストールされているコンピューターにて、有効なライセンスとともに、Altova のSchemaAgent Client 製品をインストールする必要があります。

24.7.10 SchemaAgent Server から切断

「**SchemaAgent Server から切断**」コマンドは、SchemaAgent Server への接続が正常に行われている際に有効になります。このコマンドを選択することで、XMLSpy がSchemaAgent Server から切断されます。

24.7.11 SchemaAgent で表示

「**SchemaAgent で表示**」メニューアイテムにより、アクティブなスキーマ（そしてオプションとして、リンクされているスキーマ）をAltova のSchemaAgent にて表示することができます。SchemaAgent の機能を使用するには、XMLSpy が動作しているコンピューターにこの製品をインストールする必要があります。スキーマがSchemaAgent のSchemaAgent Design にて新たに開かれます。

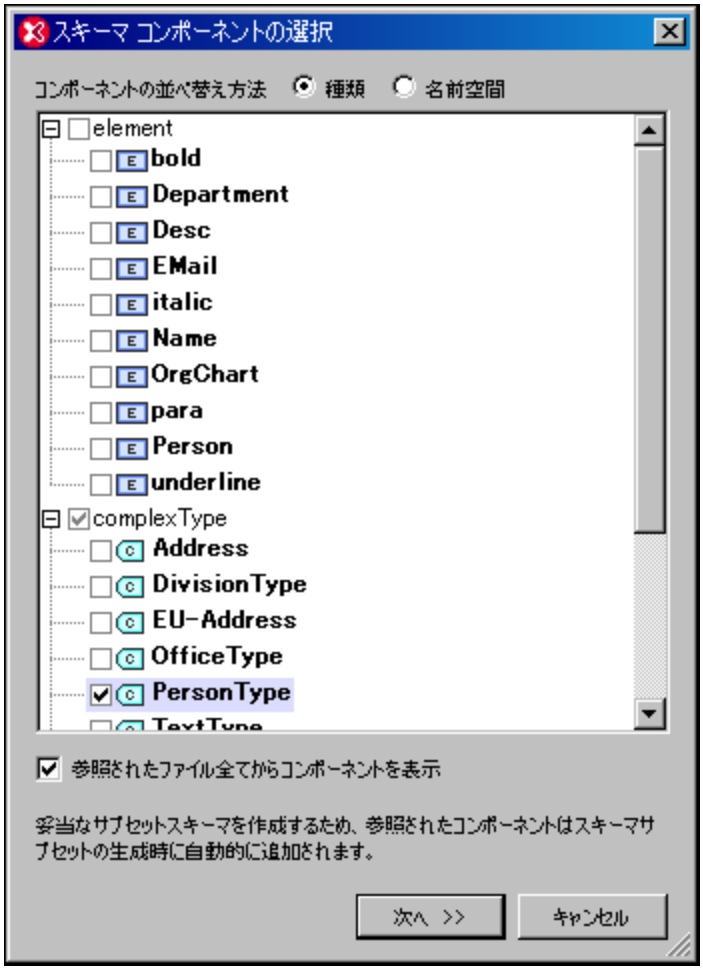
SchemaAgent で表示メニューアイテムにマウスオーバーすると、どのスキーマをSchemaAgent で表示するか選択するためのオプションがサブメニューにて表示されます。これらのオプションについては、このユーザーマニュアルのDTD/XML スキーマセクション以下にある SchemaAgent 298 を参照ください。

24.7.12 SchemaAgent 検証

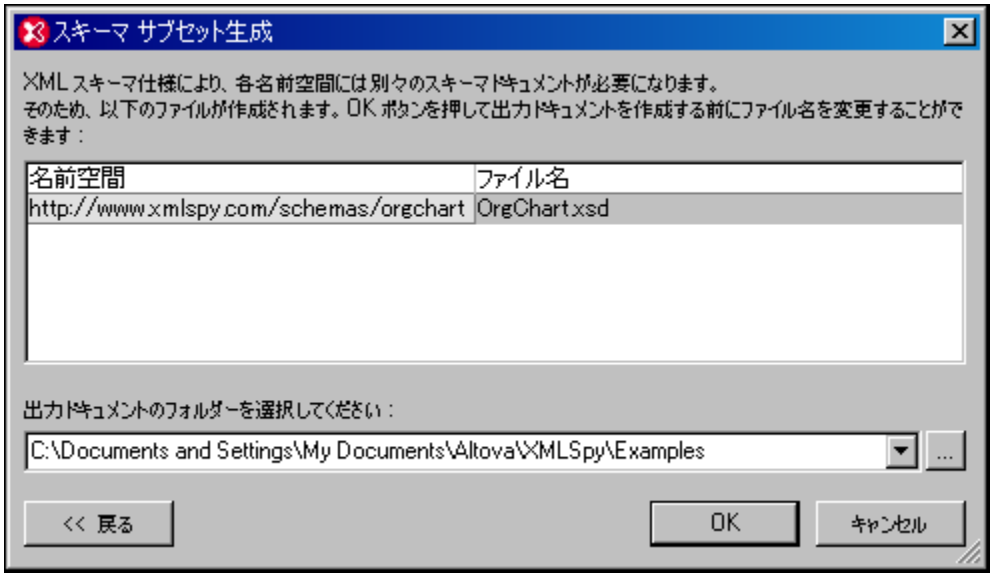
「**SchemaAgent 検証**」コマンドにより、現在アクティブなスキーマと、現在アクティブなスキーマに関連したスキーマを検証することができます。この機能に関する詳細は、このユーザーマニュアルのスキーマビューセクション以下にあるSchemaAgent 検証 298 のセクションを参照ください。

24.7.13　スキーマサブセットの作成

「**スキーマサブセットの作成**」コマンドにより、スキーマコンポーネントの選択ダイアログが表示されます（以下のスクリーンショットを参照）。このダイアログでは、単一のスキーマサブセットとして作成したいコンポーネントにチェックを入れて、**次へ**をクリックします。ペイン以下にあるチェックボックスにより、参照されている全てのファイルを表示、選択することができます。



表示されるスキーマサブセット生成ダイアログにて、スキーマサブセットパッケージのファイル名を入力します（以下のスクリーンショットを参照）。新たに作成されるスキーマサブセットファイルが保存される場所も指定する必要があります。作成されるコンポーネントが、オリジナルのスキーマにおいてインポートされたコンポーネントである場合は、スキーマサブセットパッケージを複数のファイルにより構成することもできます。異なるスキーマファイルが、スキーマサブセット内の各名前空間に対して作成されます。ダイアログペインに表示されるファイル名は、デフォルトでオリジナルのファイル名となっていますが、オリジナルのファイルを上書きすることはできないので、オリジナルのファイルと同じ場所に保存する場合は、新しいファイル名を入力してください。



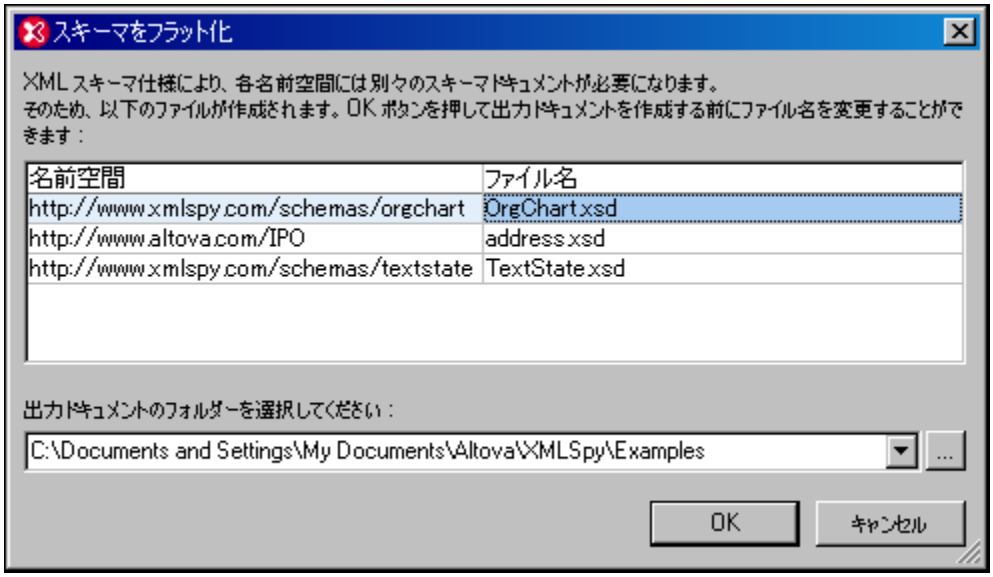
「OK」をクリックすると、アクティブなファイルに対応した名前空間を持つスキーマサブセットファイルが、スキーマビューで開かれます。パッケージ内にあるその他のファイルは、作成されますが、スキーマビューで開かれません。

24.7.14 スキーマのフラット化

スキーマビューに表示されているアクティブなスキーマのフラット化は、(i) インクルードされている全てのスキーマをグローバルコンポーネントとしてアクティブなスキーマに追加した後、(ii) インクルードされたスキーマを削除する、という処理により構成されます。

アクティブなスキーマをフラット化するには、「**スキーマ設計 | スキーマのフラット化**」コマンドを選択します。この操作によりスキーマをフラット化ダイアログが表示され、フラット化されたスキーマにおける各名前空間に対応したファイルの名前が表示されます。これらのデフォルト名は、オリジナルのファイル名と同じものですが、オリジナルのファイルを上書きすることはできないので、オリジナルのファイルと同じ場所に保存する場合は、新しいファイル名を入力してください。参照により、フラット化されたスキーマならびに関連するファイルが保存されるフォルダーを指定することができます。

(C) 2015-2021 Altova GmbH



「OK」をクリックすると、フラット化されたスキーマがスキーマビューにて開かれます。

24.8 XSL/XQuery メニュー

XSL 変換言語を使えば、どのようにXML ドキュメントが他のXML ドキュメントやテキストファイルへ変換されるかを指定することができます。XSLT ドキュメントにより生成されるXML ドキュメントの1つに、PDF の生成処理を行うために使用されるFO ドキュメントがあります。XMLSpy には(XSLT 1.0, XSLT 2.0, and XSLT 3.0 のための) XSLT プロセッサーが内蔵されており、使用中のシステムにある FO プロセッサーと連携することで、XML ファイルから様々な形式の出力を生成することができます。XMLSpy インターフェースから、直接 FO プロセッサーを使用するには、FO プロセッサーの場所をオプションダイアログ(「ツール」オプション」1038)のXSL セクションにて指定する必要があります。

XMLSpy にはXQuery エンジンも内蔵されており、(XML ドキュメントへの参照無しでも)XQuery ドキュメントを実行することができます。

上記の変換を行うためのコマンドは、全て **XSL/XQuery** メニューに収められています。更に、このメニューからはAltova XSLT/XQuery デバッガーを使用するためのコマンドへもアクセスすることができます。

(C) 2015-2021 Altova GmbH

XSL 変換(X)	F10
XSL:FO 変換(F)	Ctrl+F10
XSL パラメーター/XQuery 変数(P)...	
XQuery/Update を実行(Q)	Alt+F10
XSL の割り当て...	
XSL:FO の割り当て...	
サンプル XML ファイルの割り当て...	
XSL へ移動	
デバッガーの開始/実行	Alt+F11
デバッガーの停止	
デバッガーの再起動	
デバッグ セッションの終了	
ステップ イン	F11
ステップ アウト	Shift+F11
ステップ オーバー	Ctrl+F11
現在実行中のノードを表示	
ブレークポイントの挿入/削除	F9
トレース ポイントの挿入/削除	Shift+F9
ブレーク ポイントの有効化/無効化	Ctrl+F9
トレース ポイントの有効化/無効化	Ctrl+Shift+F9
ブレーク ポイント/トレースポイント...	
デバッグ ウィンドウ	▶
デバッグ の設定...	

24.8.1 XSL 変換

F10

「XSL/XQuery | XSL 変換」コマンドにより、割り当てられたXSLT スタイルシートを使ってXMLドキュメントの変換を行うことができます。変換は、内蔵されている適切なAltova XSLT エンジン(XSLT 1.0 スタイルシートに対してはAltova XSLT 1.0 エンジンを、XSLT 2.0 スタイルシートに対してはAltova XSLT 2.0 エンジンを使用)、Microsoft から提供されているMSXML モジュール、または外部のXSLT プロセッサーを使用することで行うことができます。このコマンドにより使用されるプロセッサーは、オプションダイアログ(「ツール | オプション」)の XSL 1036 セクションにて指定することができます。

XMLドキュメントにXSLT スタイルシートへの参照が含まれている場合、このスタイルシートが変換に使用されます(XSL の割り当て 894 コマンドを使うことで、XSLT スタイルシートをXMLドキュメントへ割り当てることができます。XMLドキュメントがプロジェクトに含まれている場合 プロジェクトプロパティ 808 ダイアログにて、フォルダーごとにXSLT スタイルシートを指定することができます。変換を行うプロジェクトフォルダーやファイルを右クリックして「XSL 変換」を選択してください)。XSLT スタイルシートがXML ファイルに割り当てられていない場合、使用するXSLT スタイルシートを指定するよう促されます。ファイルの選択は(参照 746 ボタンをクリックして)グローバルリソースやURL、(ウィンドウボタンをクリックして) XMLSpy にて既に開かれているファイルからも行うことができます。

RaptorXML 2021 を使用した自動検証

RaptorXML は、XML 検証、XSLT 変換、とXQuery 変換のためのAltova のスタンドアロンアプリケーションです。Java プログラムと.NET 内のCOM インターフェイスを使用して、アプリケーションコマンドラインから使用することができます。このため、RaptorXML を使用してXSLT 変換タスクを実行することができます。例えば、ドキュメントのセット上でXSLT 変換を実行するためにRaptorXML を呼び出すバッチファイルを作成し、テキストファイルに出力を送ることができます。詳細に関しては、RaptorXML ドキュメンテーション を参照ください。

ZIP ファイルへの変換

変換の結果を .docx 拡張子を持ったOpen Office XML (OOXML) やZIP ファイルに収める場合は、出力ファイルのファイルパスに以下のような ZIP プロトコルを指定する必要があります:

```
filename.zip|zip/filename.xxx

filename.docx|zip/filename.xxx
```

メモ: 変換を行う前に、ディレクトリ構造の作成を行わなければならない場合もあります。Open Office XML アーカイブ形式で変換を行う場合、(例えば .docx のような)トップレベルの OOXML ファイルを作成するために、アーカイブファイルの ZIP 化を行う必要があります。

24.8.2 XSL-FO 変換

Ctrl+F10

FO は、印刷することを前提にしたドキュメントを記述するためのXML フォーマットです。Apache XML プロジェクトのFOP のようなFO プロセッサーでは、FO ファイルを入力することでPDF を出力として生成することができます。そのため、XMLドキュメントからPDF を生成するには2つのステップを踏むことになります。

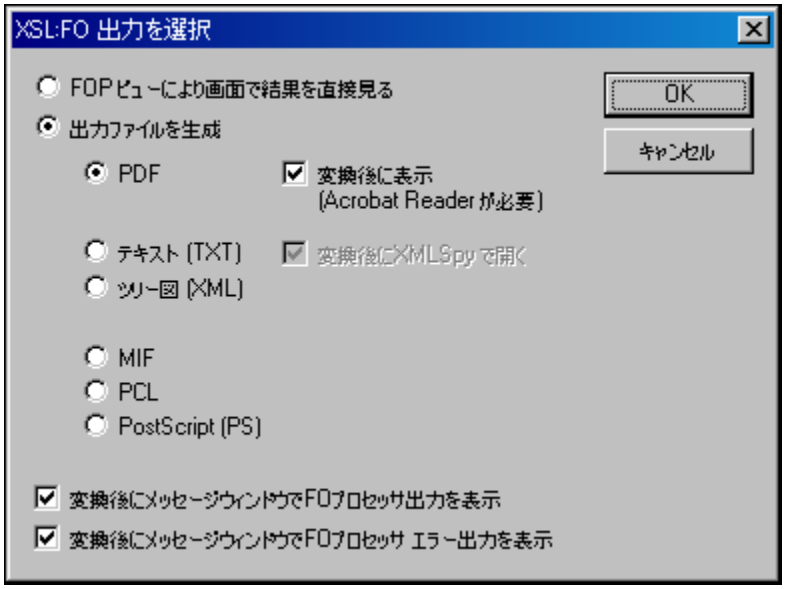
1. XSLT (または XSL-FO)スタイルシートを使うことで XMLドキュメントを FO ドキュメントへ変換します。
2. FO プロセッサーにより FO ドキュメントを処理することで、PDF(またはその他の出力)を生成します。

「XSLT/XQuery | XSL:FO 変換」コマンドによりXMLドキュメントやFOドキュメントをPDFに変換します。

- XSL:FO 変換コマンドをソース XML ドキュメントに対して実行すると、上にある両方のステップが順に処理されます。FO への変換に必要な XSLT(または XSL-FO)スタイルシートが XML ドキュメントから参照されていない場合、変換に使用するスタイルシートを指定する必要があります(以下のスクリーンショットを参照)。ファイルの選択は(参照[746] ボタンをクリックして)グローバルリソースやURL、(ウィンドウボタンをクリックして) XMLSpy にて既に開かれているファイルからも行うことができます。XML からXSL-FO への変換が、オプションダイアログ(「ツール | オプション」)のXSL タブ[1036] にて指定されたXSLT プロセッサーにより行われます。デフォルトでは、XMLSpy XSLT プロセッサーが選択されます。結果として生成されるFO ドキュメントが、オプションダイアログ(「ツール | オプション」)のXSL タブ[1036] にて指定されたFO プロセッサーにより変換されます。
- XSL:FO 変換コマンドがFO ドキュメントに対して実行されると、ドキュメントがオプションダイアログ(「ツール | オプション」)のXSL タブ[1036]にて指定されたFO プロセッサーにて処理されます。

XSL:FO 変換出力

XSL:FO 変換コマンドを選択すると、XSL:FO 出力を選択ダイアログが表示されます(以下のスクリーンショットを参照)。アクティブなドキュメントがXSLT 割り当てを持たないXMLドキュメントの場合、XSLT ファイルの選択を促されます。



FO プロセッサーの出力はFOP ビューアーを使用することで、画面上にて直接確認する他にも、以下のフォーマットにより出力ファイルを生成することができます:PDF、テキスト、XML エリアツリー、MIF PCL、またはPostScript。FO プロセッサーから得られたメッセージ出力するオプションにより、(i) プロセッサーの標準出力メッセージや(ii) プロセッサーのエラーメッセージをメッセージウィンドウにて表示することができます。これら2つのオプションを有効にするには、ダイアログ下部にある対応するチェックボックスにチェックを入れてください。

メモ:

- Apache XML プロジェクト のFOP プロセッサーをインストールするオプションを解除していない限り、次のフォルダーにインストールされます:`C:\ProgramData\Altova\SharedBetweenVersions`. インストールされると、パスは自動的にオプションダイアログ(ツール | オプション)内のXSL タブ[1036] に使用するFO プロセッサーとして入力されます。使用するFO プロセッサーにパスを設定することができます。

- XSL：変換コマンドは、メインウィンドウにてアクティブとなっているファイルだけではなく、アクティブなプロジェクトにて選択することのできるファイルやフォルダーに対しても使用することができます。右クリックで表示されるコンテキストメニューから**XSL：FO 変換**を選択してください。選択されたプロジェクトフォルダーに割り当てられたXSLT スタイルシートが使用されます。

24.8.3 XSL パラメーター / XQuery 実行

「**XSL/XQuery | XSL パラメーター/XQuery 変数**」コマンドにより、XSLT 入力パラメーター/XQuery 外部変数ダイアログが表示されます（*スクリーンショットを参照*）。XSLT スタイルシートにパスするパラメーターの名前や、XQuery ドキュメントへパスするXQuery 変数の名前と、対応する値を入力します。XMLSpy においてこれらのパラメーターは以下のように使用されます：

- XSL/XQuery メニューの**XSL 変換**コマンドがXML ドキュメントの変換に使用される際、ダイアログ内にて保存されたパラメーター値が選択されたXSLT ドキュメントへ渡され、変換に使用されます。
- XSL/XQuery メニューの**XQuery 実行**コマンドがXQuery ドキュメントの処理に使用される際、ダイアログにて現在保存されているXQuery 外部変数の値がXQuery ドキュメントに渡され、実行に使用されます。

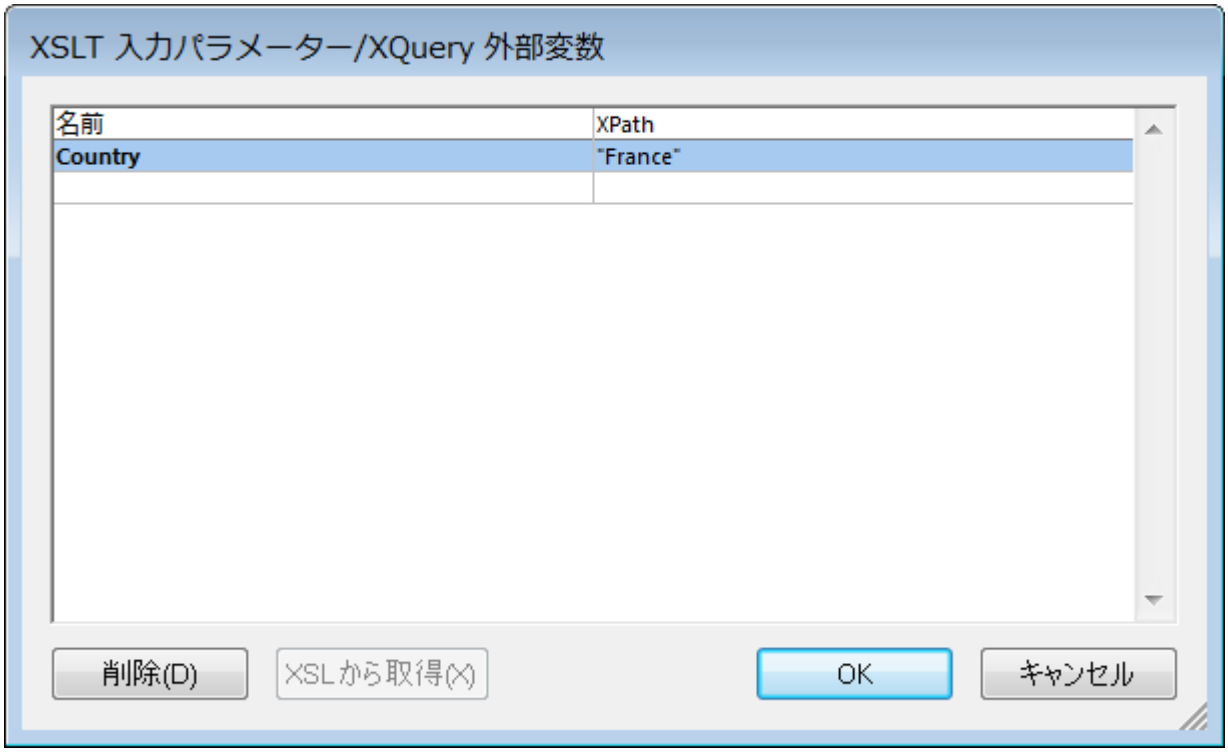
メモ XSLT 入力パラメーター/XQuery 外部変数ダイアログにて入力されたパラメーターや変数は、内蔵のAltova XSLT エンジンに対してのみ使用されます。このため、MSXML、または、構成済みの外部エンジンを使用する場合、これらの値はエンジンにパスされません。

メモ XSLT 入力パラメーター/XQuery 外部変数 ダイアログ内でXSLT パラメーター、または、外部 XQuery 変数 を定義することはエラーではありませんが、XSLT/XQuery ドキュメント、または、変換内では使用されません。

XSLT パラメーターの使用

パラメーターに対して入力した値は引用句無しのXPath 条件式や、引用句無しのテキスト文字列となります。アクティブなドキュメントがXSLT ドキュメントの場合、**XSL から取得**ボタンが有効になります。このボタンをクリックすることで、XSLT にて宣言されているパラメーターが、デフォルト値とともにダイアログに入力されます。この操作により、宣言されたパラメーターを素早く入力し、デフォルト値を必要に応じて変更することができます。

(C) 2015-2021 Altova GmbH



メモ パラメーター値がダイアログに入力されると、明示的に削除、または、アプリケーションが再起動されるまですべての変換で使用されます。ダイアログ内に入力されるパラメーターはそのセッションのアプリケーションレベルで指定され、この時点からIDE を使用して実行される各変換のためのそれぞれのXSLT ドキュメントためにパスされます。これは以下を意味します:

- パラメーターは特定のドキュメントに割り当てられているのではありません。
- ダイアログにて入力されたパラメーターは、XMLSpy の終了時に削除されます。

XSLT パラメーターの使用例

国名とそれぞれの首都を含むXML ドキュメントは以下のとおりです:

```
<document>
   <countries>
      <country name="USA" capital="Washington DC"/>
      <country name="UK" capital="London"/>
      <country name="France" capital="Paris"/>
      <country name="Russia" capital="Moscow"/>
      <country name="China" capital="Beijing"/>
   </countries>
</document>
```

次のXSLT ドキュメントは、XML ファイルから国名と首都を表示するXML ドキュメントを生成します。**country** という名前を持つパラメーターの値として名前を入力することにより国名は選択されます(*以下では黄色のハイライトにより表示されています*)。

```
<xsl:stylesheet version="2.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
   <xsl:param name="country" select="'USA'"/>
   <xsl:template match="countries">
      <xsl:for-each select="country[@name=$country]">
         <country>
```

```
            <name><xsl:value-of select="$country"/></name>
            <capital><xsl:value-of select="@capital"/></capital>
        </country>
    </xsl:for-each>
  </xsl:template>
</xsl:stylesheet>
```

このXSLTドキュメントが上記のXMLドキュメントで実行されると、結果は以下のようになります:

```
<country><name>USA</name><capital>Washington DC</capital></country>
```

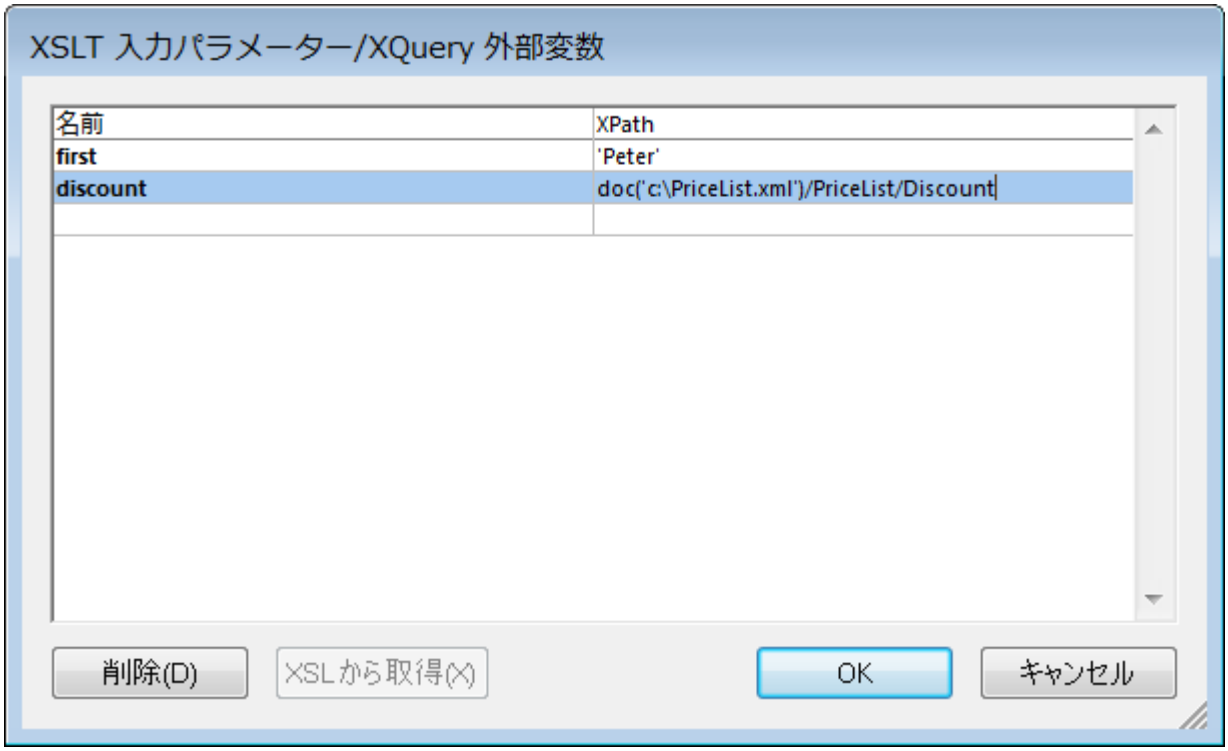
XSLT 入力パラメーター/XQuery 外部変数 ダイアログ内で **country** という名前のパラメーターが作成され、値が与えられると、新規のパラメーターの値は、XSLT 変換内のパラメーターにパスされます(*上のスクリーンショット参照*)。この値は変換のためにXSLT スタイルシート内のパラメーター **country** にパスされます。このようにして、異なるパラメーターへ異なる値をランタイムにパスすることができます。

メモ

- XSL:FO 変換コマンド(「XSL/XQuery | XSL:FO 変換」)を使用する場合、ダイアログにて入力された値はスタイルシートへ渡されません。これらのパラメーターをPDF 出力にて使用するには、まずXSLT 変換コマンド(「**XSL/XQuery | XSL 変換**」)を使用して XML から FO ドキュメントへの変換を行い、その後「XSL:FO 変換」コマンド(XSL/XQuery | XSL:FO T変換)によりFO からPDF への変換を行う必要があります。
- 内蔵されているAltova XSLT エンジン以外のXSLT プロセッサーを使用する場合、ダイアログにて入力されたパラメーターは外部プロセッサーへ渡されません。

外部 XQuery 変数の使用

外部 XQuery 変数に対して入力する値は、引用句無しのXPath 条件式か、引用句により分離されたテキスト文字列となります。外部変数のデータ型は、XQuery ドキュメントの変数宣言にて指定されます。



(C) 2015-2021 Altova GmbH

メモ　外部 XQuery 変数がダイアログにて一度入力されると、そのエントリーが明示的に削除されるかアプリケーションが再起動されるまで、その後の実行でもそれらの変数が使用され続けます。ダイアログにて入力された変数はアプリケーションレベルで指定されており、その後行われる実行のたびに関連する XQuery ドキュメントへ渡されます。これは以下を意味します:

- 変数は特定のドキュメントに割り当てられているのではありません。
- ダイアログにて入力された変数は、アプリケーション（XMLSpy）の終了時に削除されます。

外部 XQuery 変数の使用例

以下の例では、変数 `$first` が XQuery ドキュメントにて宣言され、FLW OR ステートメントの return 句にて使用されます:

```
xquery version "1.0";
declare variable $first as xs:string external;
let $last := "Jones"
return concat($first, " ", $last )
```

（XSLT 入力パラメーター/XQuery 外部変数ダイアログにて入力された）外部変数に `Peter` がセットされている場合、XQuery により `Peter Jones` という値が返されます。以下の点に注意してください:

- XQuery ドキュメントの変数宣言にある `external` キーワードにより、この変数が外部変数であるということが認識されます。
- 静的な変数の型定義はオプションです。変数の宣言時に変数のデータ型が定義されていない場合、変数の値は `xs:untypedAtomic` に割り当てられます。
- XQuery ドキュメントにて外部変数が宣言されているが、その変数名に対して外部変数が与えられない場合、エラーとなります。
- 外部変数が宣言され、XSLT 入力パラメーター/XQuery 外部変数ダイアログにて入力された場合、実行されるのは XQuery ドキュメントのスコープ内となります。XQuery ドキュメントにて新たな変数が同じ名前で宣言されている場合、その変数によりスコープ内の外部変数が一時的にオーバーライドされます。例えば、以下にある XQuery ドキュメントでは、（外部変数により `$first` に対して `Peter` という値が渡されているにも関わらず）`Paul Jones` という値が返されます。

```
xquery version "1.0";
declare variable $first as xs:string external;
let $first := "Paul"
let $last := "Jones"
return concat($first, " ", $last )
```

24.8.4　XQuery/Update 実行

XSL/XQuery | XQuery/Update 実行 コマンドにより XQuery（1.0/3.1）または XQuery Update（1.0/3.0）ドキュメントが実行されます。選択された XQuery または XQuery Update ファイルにより、XQuery 実行 または XQuery 更新が実行されます。XMLSpy は、ドキュメントの型（XQuery または XQuery Update）を（オプションダイアログのファイル型 1018 セクション内で定義された）ファイル型との関連 1018 をベースにして認識します。

使用する XQuery エンジン（1.0 または 3.1）は、ドキュメント内のバージョンの宣言をベースに自動的に選択されます。ドキュメント内にバージョン宣言が存在しない場合、オプションダイアログの XQuery タブ 1038 内で指定されているデフォルトのバージョンが使用されます。XQuery、XQuery Update、または XML ファイルがアクティブな場合、**XQuery/Update 実行** コマンドが呼び出されます。XML ファイルから呼び出されると、XML ファイルと関連付けられる XQuery ファイルを要求するダイアログが開かれます。グローバルリソース、または、URL（参照 746 ボタンをクリックして）または ファイルを選択することにより、（ウィンドウ ボタンをクリックして）XMLSpy 内のウィンドウで開くことができます。

メモ:　プロジェクトウィンドウ 105 アイテムのコンテキストメニューでコマンドを使用することができます。

RaptorXML 2021 を使用した自動検証

RaptorXML は、XML 検証、XSLT 変換、とXQuery 変換のためのAltova のスタンドアロンアプリケーションです。Java プログラムと.NET 内のCOM インターフェイスを使用して、アプリケーションコマンドラインから使用することができます。このため、RaptorXML を使用してXQuery 実行タスクを実行することができます。例えば、ドキュメントのセット上でXQuery を実行するためにRaptorXML を呼び出すバッチファイルを作成し、テキストファイルに出力を送ることができます。詳細に関しては、RaptorXML ドキュメンテーション を参照してください。

24.8.5 XSL の割り当て

「XSL/XQuery | XSL の割り当て...」コマンドにより、XSLT スタイルシートがXML ドキュメントへ割り当てられます。このコマンドをクリックすることで、目的のXSLT ファイルを指定するためのダイアログが表示されます。ファイルの選択は(参照 746 ボタンをクリックして) グローバルリソースやURL、(ウィンドウボタンをクリックして) XMLSpy にて既に開かれているファイルからも行うことができます。

XML ドキュメントにxml-stylesheet 処理命令が挿入されます::

```
<?xml-stylesheet type="text/xsl" href="C:¥workarea¥recursion¥recursion.xslt"?>
```

メモ:...に対する相対パスにするチェックボックスを有効にすることで、ファイルへのパスを相対パスにすることができます。

24.8.6 XSL-FO の割り当て

「XSL/XQuery | XSL:FO の割り当て」コマンドによりFO ドキュメントへの変換を行うためのXSLT スタイルシートがXML ドキュメントへ割り当てられます。このコマンドにより、XSL またはXSLT ファイルを指定するためのダイアログが表示され、適切な処理命令がXML ドキュメントに挿入されます。

...に対する相対パスにするチェックボックスを有効にすることで、ファイルへのパスを相対パスにすることができます。ファイルの選択は(参照 746 ボタンをクリックして) グローバルリソースやURL、(ウィンドウボタンをクリックして) XMLSpy にて既に開かれているファイルからも行うことができます。

メモ:XML ドキュメントには2つのXSLT ファイルを割り当てることができます。1つが標準的な XSLT 変換のためのもので、もう1つがFO へのXSLT 変換で使用されます。

24.8.7 サンプル XML ファイルの割り当て

「XSL/XQuery | サンプルXML ファイルの割り当て」コマンドによりXML ファイルをXSLT ドキュメントへ割り当てます。このコマンドにより、XSLT ファイルにてXSL 変換がなされたときに使用されるXML ファイルを指定した処理命令が挿入されます:

<?altova_samplexml C:¥workarea¥html2xml¥article.xml?>

メモ:「…に対する相対パスにする」チェックボックスを有効にすることで、ファイルへのパスを相対パスにすることができます。ファイルの選択は（参照 746 ボタンをクリックして）グローバルリソースやURL、（ウィンドウボタンをクリックして）XMLSpy にて既に開かれているファイルからも行うことができます。

24.8.8 XSL へ移動

「XSL/XQuery | XSL へ移動」コマンドにより、割り当てられているXSLT ドキュメントが開かれます。XML ドキュメントに、以下のような（XSLT 割り当てを行う）処理命令が含まれている場合、

```
<?xml-stylesheet type="text/xsl" href="Company.xsl"?>
```

「XSL へ移動」コマンドによりXSLT ドキュメントがXMLSpy にて開かれます。

24.8.9 デバッガーの開始/実行

Alt+F11

「XSL/XQuery | デバッガーの開始/実行」コマンドによりXSLT/XQuery ドキュメントの処理が開始または再開されます。ブレークポイントがセットされている場合、そこで処理が停止します。トレースポイントがセットされている場合、ステートメントの終了ノードに到着した時点で、それらステートメントの出力がトレースウィンドウに表示されます。デバッガーセッションが開始されていない場合、このボタンによりセッションが開始され、ファイルにある最初の処理ノードにて停止します。セッションが既に開始されている場合、XSLT/XQuery ドキュメントの最後、または次のブレークポイントまで処理が行われます。

24.8.10 デバッガーの停止

「XSL/XQuery | デバッガーの停止」コマンドによりデバッガーが停止します。デバッガーセッションが動作している中でデバッガーを終了するのとは少し違い、デバッグセッションの最中にドキュメントの編集を行ったり、同じデバッグセッションにて別のファイルを使用する様な場合に便利な機能となります。デバッガーを終了させると、デバッガーを再度開始して、XSLT/XQuery ドキュメントの最初からやり直す必要があります。

24.8.11 デバッガーの再起動

「XSL/XQuery | デバッガーの再起動」コマンドにより、出力ウィンドウの内容がクリアされ、現在選択されているファイルにてデバッグセッションが再度開始されます。

24.8.12 デバッグセッションの終了

「XSL/XQuery | デバッグセッションの終了」コマンドによりデバッグセッションが終了され、デバッグセッションが開始される以前にアクティブであった通常のXMLSpy ビューに切り替わります。XSLT/XQuery デバッガー設定 362 ダイアログでは、デバッグセッションにて開かれていた出力ドキュメントを開かれたままにするかどうかを設定することができます。

24.8.13 ステップイン

F11

「XSL/XQuery | ステップイン」コマンドにより、スタイルシートにある全てのノードならびにXPath 条件式の内部を処理します。このコマンドは停止されているデバッガーを再度開始するためにも使用することができます。

24.8.14 ステップアウト

Shift+F11

「XSL/XQuery | ステップアウト」コマンドにより、親ノードから見た次の兄弟ノードに移動するか親ノードから見た1つ上のレベルにある次のノードに移動します。

(C) 2015-2021 Altova GmbH

24.8.15 ステップオーバー

Ctrl+F11

「XSLT/XQuery | **ステップオーバー**」コマンドにより、現在のノードから同じレベルにある次のノードに移動するか、現在のノードより先上のレベルにある次のノードへ移動します。このコマンドは停止されているデバッガーを再度開始するためにも使用することができます。

24.8.16 現在実行中のノードを表示

「XSL/XQuery | **現在実行中のノードを表示**」コマンドにより、XSLT/XQuery ドキュメント内にある現在実行中のノードならびにXML ドキュメント内にある対応するコンテキストノードを表示/選択します。このコマンドはXSLT スタイルシートやXML ファイルのコードを表示/マーキングするために他のタブをクリックした後、それまで処理を行っていた場所に戻る際に便利な機能になります。

24.8.17 ブレークポイントの挿入/削除

F9

「XSL/XQuery | **ブレークポイントの挿入/削除**」コマンドにより、現在のカーソル位置にブレークポイントが挿入、または既にあるブレークポイントが削除されます。インラインブレークポイントはXSLT/XQuery やXML ドキュメントの両方で定義することができ、処理がどこで停止するかを決定することができます。赤色の点線がブレークポイントをセットしたノードの上に表示されます。終了ノードにブレークポイントをセットすることはできず、XSLT ドキュメントの属性に対するブレークポイントは無視されます。このコマンドは右クリックして表示されるコマンドからも利用することができます。

24.8.18 トレースポイントの挿入/削除

Shift+F9

「XSL/XQuery | **トレースポイントの挿入/削除**」コマンドにより、XSLT/XQuery ドキュメント内のカーソル位置にトレースポイントが挿入、または削除されます。デバッグ中にトレースポイントがセットされたステートメントの終了ノードにたどり着くと、そのステートメントの結果がトレースウィンドウに表示されます。青色の点線がトレースポイントをセットしたノードの上に表示されます。終了ノードにトレースポイントをセットすることはできません。このコマンドは右クリックして表示されるコンテキストメニューからも利用することができます。

24.8.19 ブレークポイントの有効化/無効化

Ctrl+F9

「XSL/XQuery | **ブレークポイントの有効化/無効化**」コマンドにより、既に定義されたブレークポイントを有効/無効化することができます。ブレークポイントが無効化されると、赤色のブレークポイントが灰色で表示されます。デバッガーは無効化されたブレークポイントでは停止しません。ノード名までカーソルを移動させ、**ブレークポイントの有効/無効化**コマンドをクリックして、ブレークポイントを有効/無効化することができます。このコマンドは右クリックにより表示されるコンテキストメニューからも利用できます。

24.8.20 トレースポイントの有効化/無効化

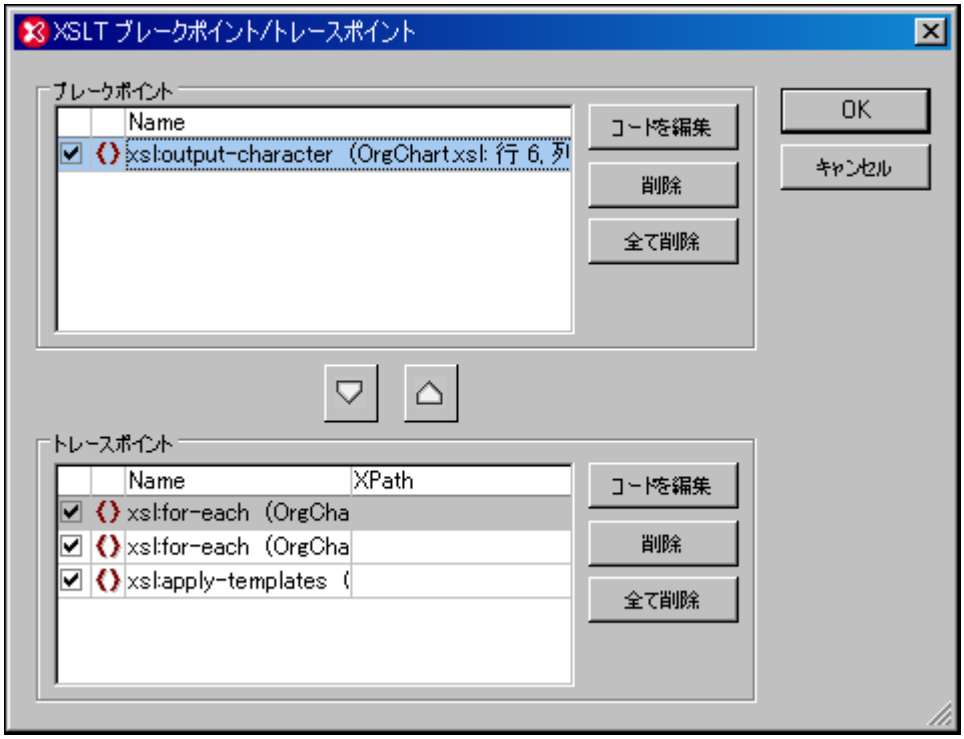
Ctrl+Shift+F9

「XSL/XQuery | **トレースポイントの有効化/無効化**」コマンドにより、既に定義されたトレースポイントを有効/無効化することができます。トレースポイントが無効化されると、青色のトレースポイントが灰色で表示されます。無効化されたトレースポイントにおいて出力は行われません。ノード名までカーソルを移動させ、**トレースポイントの有効/無効化**コマンドをクリックして、トレースポイントを有効/無効化することができます。このコマンドは右クリックにより表示されるコンテキストメニューからも利用できます。

24.8.21 ブレークポイント/トレースポイント

「XSL/XQuery | **ブレークポイント/トレースポイント**」により、XSLT ブレークポイント/トレースポイントダイアログが表示され、現在のデバッグセッションにおいて全てのファイルで定義されているブレークポイントやトレースポイントが(無効になっているブレークポイントやトレースポイントも含めて)表示されます。

(C) 2015-2021 Altova GmbH



各ブレークポイントやトレースポイントの隣にあるチェックボックスは、それらが有効になっているかどうかを表します。「**削除**」ボタンをクリックすることで、ハイライトされているブレークポイントやトレースポイントを削除することができます。「**全て削除**」ボタンをクリックすることで、表示されている全てのブレークポイントまたはトレースポイントが削除されます。「**コードを編集**」ボタンをクリックすると、ファイル内にあるブレークポイント/トレースポイントへ直接ジャンプすることができます。

下矢印 ボタンを使用することで、ハイライトされたブレークポイントを「**トレースポイント**」ペインへ、上矢印 ボタンにより、ハイライトされたトレースポイントを「**ブレークポイント**」ペインへ移動することができます。
トレースポイントペインの**XPath** カラムでは、各トレースポイントに対して XPath をセットすることができます。

24.8.22 デバッグウィンドウ

「XSL/XQuery | **デバッグウィンドウ**」コマンドのサブメニューには、XSLT/XQuery デバッガーにある様々な情報ウィンドウが用意されています。リストにある情報ウィンドウを選択することで、XSLT/XQuery デバッガーインタフェース上にあるウィンドウの表示/非表示を選択することができます。このコマンドは、デバッグセッション中にのみ使用することができます。

24.8.23 デバッグ設定

「XSL/XQuery | XSLT/XQuery **の設定**」コマンドにより、デバッグ設定ダイアログ 362 が開かれ、デバッガーのユーザーオプションを選択することができます。詳細はXSLT/XQuery デバッガー 356 セクションを参照下さい。

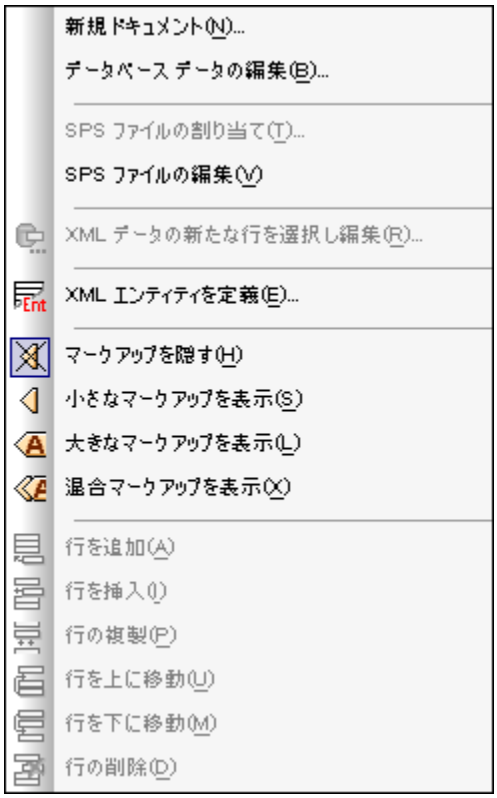
(C) 2015-2021 Altova GmbH

24.9 Authentic メニュー

Authentic View を使用することで、**Altova StyleVision** により作成された **StyleVision Power Stylesheet**（.sps ファイル）をベースにしたドキュメントの編集を行うことができます！スタイルシートにはXML ファイルを Authentic View にて視覚的に表示するための情報が収められています。表示するための情報に加えて、StyleVision Power Stylesheet によりXML ファイルへデータを書き込むこともできます。データはXSLT スタイルシートに用意されている機能により動的に処理され、その結果は Authentic View にて直ちに表示されます。

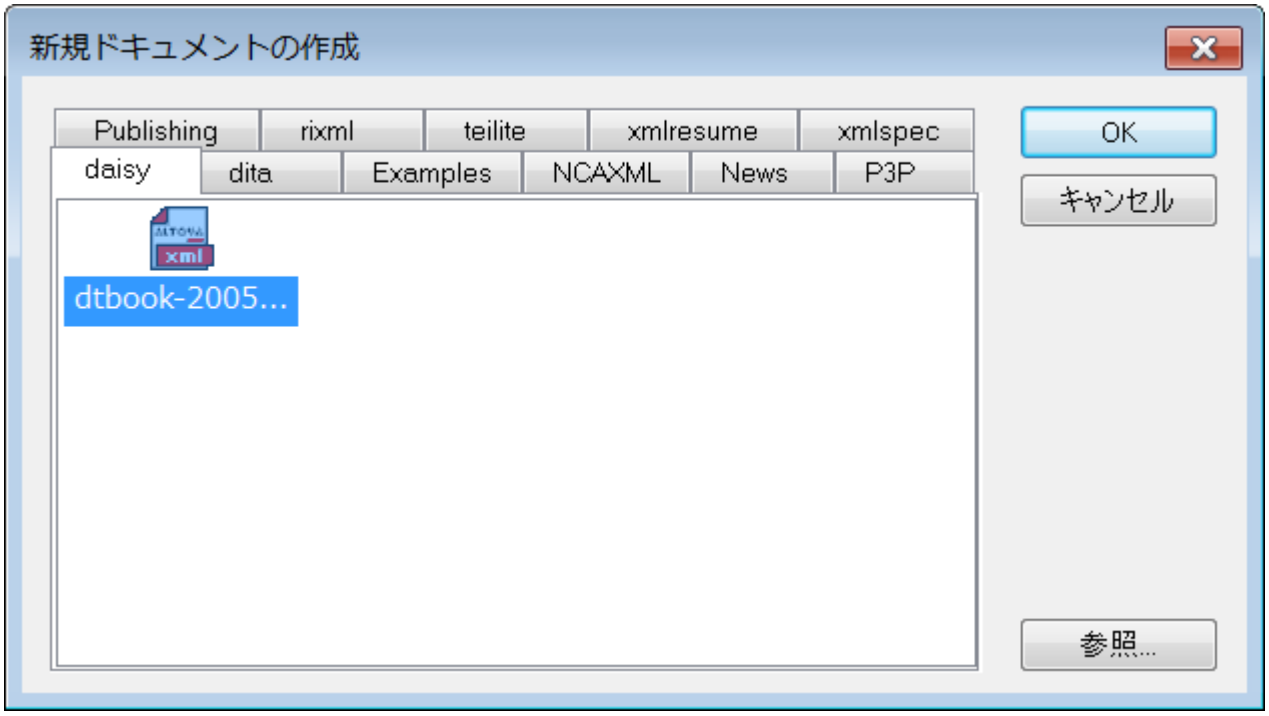
他にも StyleVision Power Stylesheet を使うことでデータベースにあるデータの編集を行うこともできます StyleVision Power Stylesheet にはデータベースへ接続するための情報が含まれており、データベースから得られたデータを Authentic View にて表示し、編集後のデータをデータベースへ書き込むことができます。

Authentic メニューコマンドにはXML ドキュメントを Authentic View にて編集するためのコマンドが収められています。のチュートリアルは、 Authentic View チュートリアル 403 セクションを参照ください。



24.9.1 新規ドキュメント

このコマンドにより Authentic View 内で新規のXML ドキュメントテンプレートを開く事ができます。XML ドキュメントテンプレートは、StyleVision Power Stylesheet (.sps ファイル)をベースにしています。新規のドキュメントの作成 ダイアログ内で StyleVision Power Stylesheet（SPS ファイル）を選択する事により開かれます（*下のスクリーンショット*）。 SPS を選択して、「OK」をクリックすると、SPS ファイルのために定義されたXML ドキュメントテンプレートが Authentic View 内で開かれます。



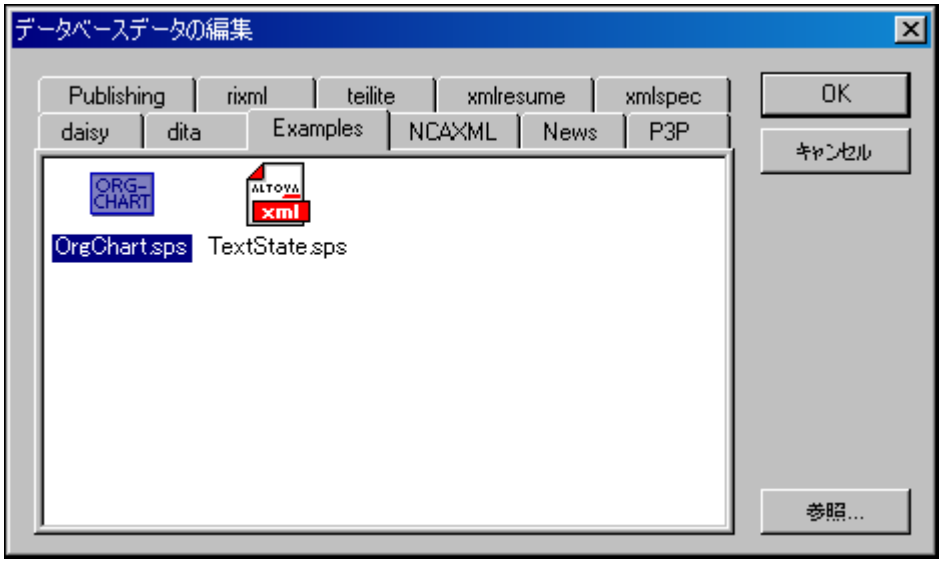
新規のドキュメントの作成 ダイアログは、一般的に使用されるDTD またはスキーマをベースにした、XMLドキュメントテンプレートの選択を提供します。またはテンプレート XML ファイルが割り当てられているカスタムメイドのSPS ファイルを参照することができます。SPS ファイルは、Altova StyleVision を使用して作成されます。アプリケーションにより、DTD またはXML スキーマをベースにしたXMLドキュメントテンプレートをデザインすることができます。StyleVision 内で必要な SPS をデザインすると、XML ファイルは、(StyleVision 内で) SPS へのテンプレート XML ファイルとして割り当てられます。XML ファイル内のデータは、XMLSpy のAuthentic View 内で開かれた新規のドキュメントテンプレートの開始データを与えます。

新規のXMLドキュメントテンプレートは、ですから、テンプレート XML ファイルとして選択されたSPS とXML ファイルのデータ内で定義されたドキュメントプレゼンテーションプロパティを持つようになります。Authentic View ユーザーは、XMLドキュメントテンプレートをグラフィカルなW YSIW YG インターフェイスで編集することができ、XMLドキュメントとして保存することができます。

24.9.2 データベースの編集

「Authentic | データベースの編集...」コマンドにより、データベースのビューをAuthentic View にて開くことができます。DB への接続方法やDB の表示方法、そしてAuthentic View にて許されている編集方法などは、StyleVision Power Stylesheet に記述されます。データベースの編集コマンドで開かれるのは、そのようなDB をベースにしたStyleVision Power Stylesheetです。このコマンドにより、DB への接続が確立され、DB のデータが(XML を介して) Authentic View にて表示されます

データベースの編集... コマンドにより、データベースデータの編集ダイアログが表示されます。



目的のSPS ファイルを選択してください。この操作によりDB への接続が行われ、DB が編集できる状態でAuthentic View に表示されます。Authentic View にて表示されているDB ビューのデザインはStyleVision Power Stylesheet に含まれています。

注意: **データベースの編集**コマンドを使ってDB をベースにしていないStyleVision Power Stylesheet や、StyleVision 2005 よりも前のStyleVision で作成されたDB ベースのStyleVision Power Stylesheet を開こうとした場合、エラーが表示されます。

メモ: StyleVision Power Stylesheet はAltova StyleVision により作成されます。

24.9.3 StyleVision スタイルシートの割り当て

StyleVision スタイルシートの割り当てコマンドにより、StyleVision Power Stylesheet が**XML ドキュメント**に割り当てられ、その XML ドキュメントをAuthentic View にて閲覧ならびに編集することができるようになります。XML ファイルに割り当てられる StyleVision Power Stylesheet は、XML ファイルがベースとしているスキーマファイルをベースにしなければなりません。

StyleVision Power Stylesheet をXML ファイルへ割り当てるには:

1. XML ファイルがアクティブな状態で、「 **Authentic** | **StyleVision スタイルシートの割り当て**...」コマンドを選択します。
2. この操作により、XML への割り当てを行う StyleVision Power Stylesheet ファイルを指定します。
3. 「**OK**」をクリックして、目的の SPS スタイルシートを XML ドキュメントに挿入します。「**...に対する相対パスにする**」チェックボックスを有効にすることで、ファイルへのパスを相対パスにすることができます。ファイルの選択は(参照 746 ボタンをクリックして) グローバルリソースやURL、(**ウィンドウ**ボタンをクリックして) XMLSpy にて既に開かれているファイルからも行うことができます。

```
<?xml version="1.0" encoding="UTF-8"?>
<?altova_sps HTML-Orgchart.sps?>
```

StyleVision Power Stylesheet がHTML-Orgchart.sps という名前で、XML ファイルと同じディレクトリ内に収められていることが分かります。

メモ: : 以前のバージョンのAltova 製品では、ターゲットかxmlspysps という名前とともに処理命令を使用しており、処理命令は<?xmlspysps HTML-Orgchart.sps?> というような形式になりました。最新のAltova 製品では、このような古いバージョンの処理命令もAuthentic View の表示に使用することができます。

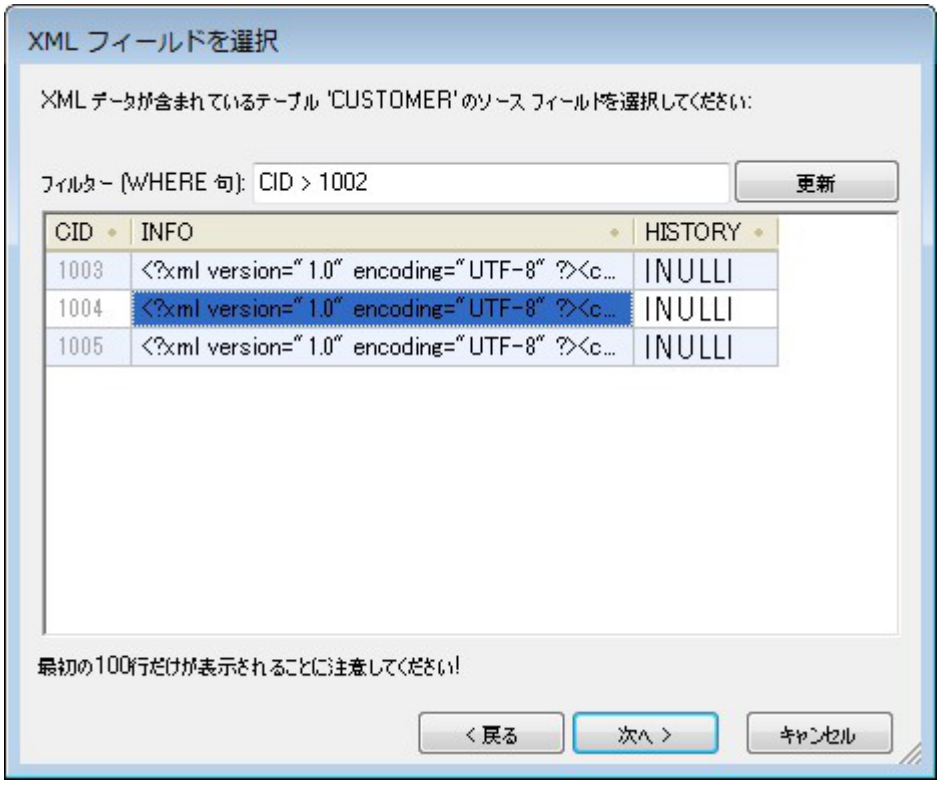
24.9.4 StyleVision スタイルシートの編集

Authentic | 編集 StyleVision スタイルシート コマンドはAuthentic View のみで使用することができ、すなわちStyleVision Power Stylesheet がXML ドキュメントに割り当てられています。StyleVision を開始し StyleVision内ですぐにStyleVision Power Stylesheet を編集することを許可します。

24.9.5 XML データの新たな行を選択し編集

「XML データの新たな行を選択し編集」コマンドにより、IBM DB2 といったXML DB にあるテーブルから新たに行を選択することができます。新たな行はAuthentic View に表示され、編集を行った後にDB への書き込みを行うことができます。

XML DB がXML データソースとして使用される場合、XML データカラムのセルに収められているXML ドキュメントがAuthentic View に表示されます。XML データの新たな行を選択し編集コマンドにより、XML カラム上の他のセル(または行)にあるXML ドキュメントを選択することができます。このコマンドを選択することで、XML フィールドの選択ダイアログが表示され、XML カラムに含まれているテーブルが表示されます。



フィルタリングによりこのテーブルに表示される内容を制限することができます。フィルタリングに使用されるのはSQL のWHERE 句です(例えば、CID>1002 といったWHERE 句の中に入る条件になります)。更新ボタンをクリックして、ダイアログの表示内容を更新することができま

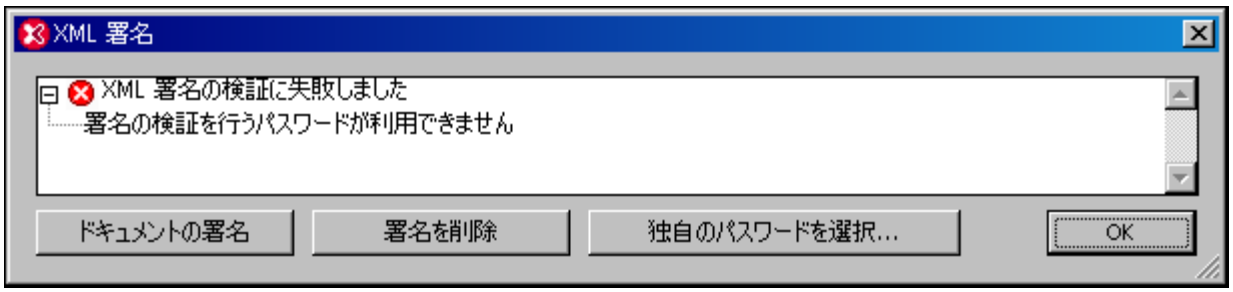
す。上のスクリーンショットではフィルタリングされた結果を見ることができます。目的のXMLドキュメントが表示されているセルを選択して**OK**をクリックします。選択されたセルのXMLドキュメントがAuthentic Viewにロードされます。

24.9.6 XML 署名

関連付けられているSPSにてXML署名が有効になっている場合、Authentic ビューで「**XML 署名**」コマンドを使用することができるようになります。Authentic ツールバーにある「**XML 署名**」ツールバーアイコン からもXML署名コマンドへアクセスすることができます。

検証と独自の証明書/パスワード

「**XML 署名**」コマンドをクリックすると、署名の検証処理が開始されます。ドキュメント内に署名が存在しない場合、その結果がXML署名ダイアログに表示され、表示されているボタンからAuthentic View ユーザーはドキュメントの署名を行うことができるようになります(以下のスクリーンショットを参照)。



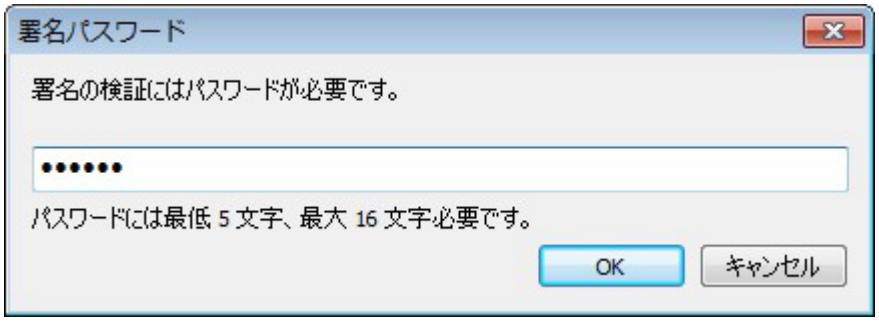
Authentic Viewにて独自の証明書/パスワードを選択することができるようにオプションが指定されている場合、「**独自の署名を選択**」または「**独自のパスワードを選択**」ボタンがダイアログにて表示されます。認証において証明書が使用されるかパスワードが使用されるかは、SPS デザイナーが署名の設定を行った際に指定されます。署名は証明書またはパスワードをベースにしたものになります。ダイアログにボタンが表示されている場合、そのボタンをクリックすることで、Authentic View が証明書を選択、またはパスワードを入力することができるようになります。Authentic View ユーザーの選択はメモリ上に保管されており、そのセッションでのみ有効になります。証明書の選択またはパスワードの入力を行った後にドキュメントやアプリケーションが閉じられると、証明書/パスワードの設定も、SPS に保存されたオリジナルの設定に戻されます。

検証ならびに認証情報

単一のドキュメントに対して検証処理が行われた場合、2つのケースが考えられます。認証情報(署名またはSPS にて)が利用できる場合、検証処理が直接実行され、その結果が表示されます(以下のスクリーンショットを参照)。

認証情報は証明書の鍵情報または署名時に使用したパスワードとなります。XML ドキュメントが署名されたときに、証明書の鍵情報が署名の内に収められるか、またパスワードをベースにした署名の場合、パスワードがSPS の中に収められるかは、SPS デザイナーにより指定されます。どちらの場合でも認証を行うことができ、検証処理はAuthentic View ユーザーからの入力を必要とすることなく直接実行されます。

もう1つのケースは、認証情報が署名の中に存在しない(証明書の場合)、またはSPS ファイルの中に存在しない(パスワードの場合)というものです。この場合、認証情報(パスワードか証明書の場所)を入力するよう求められます(以下のスクリーンショットを参照)。



24.9.7 XML エンティティの定義

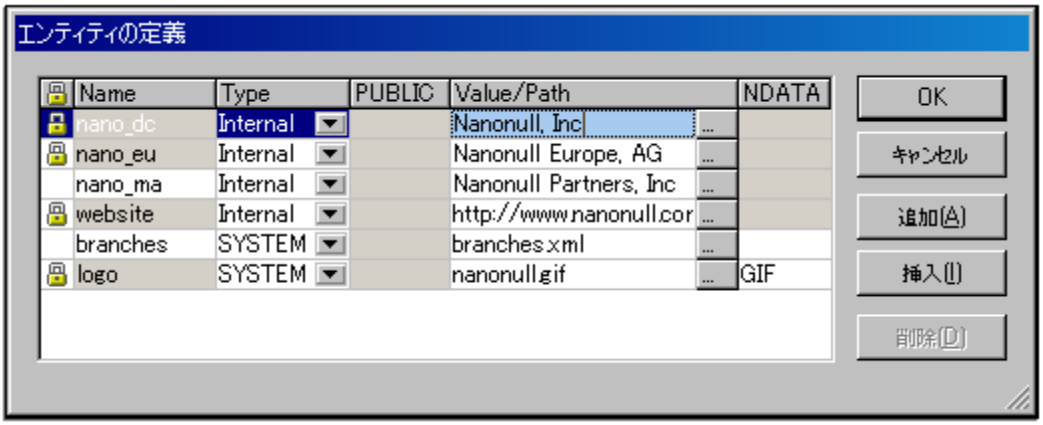
Authentic View 内で使用するエンティティを定義することができます。定義後、これらのエンティティはエンティティの入力ヘルパー内に表示され、コンテキストメニューの「**エンティティの挿入**」サブメニュー内に表示されます。エンティティの入力ヘルパー内のエンティティをダブルクリックすると、カーソル挿入ポイントにエンティティが挿入されます。

テキスト文字列、XML フラグメント、または、他の外部リソースをドキュメント内の複数の場所で使用する場合、エンティティは役にたちます。必要とされるデータのための短い名前であるエンティティを定義するには、エンティティの定義 ダイアログを使用します。エンティティを一度定義すると、そのエンティティをドキュメントの複数箇所で使用することができるようになります。この機能により、編集時間の短縮や、メンテナンスコストを大幅に軽減することができます。

ドキュメントで使用することのできるエンティティには、XML データ(テキスト文字列またはXML ドキュメントのフラグメント)の**解析対象実体**(解析対象エンティティ)と、バイナリファイル(通常はイメージや音楽などのマルチメディアオブジェクト)の様な非 XML データを扱う**解析対象外実体**(解析対象外エンティティ)と、大きく分けて2つの種類があります。各エンティティには名前と値が与えられています。解析されたエンティティの場合、エンティティはXML データのためのプレースホルダです。エンティティの値 はXML データ 自身、または、データを含む `.xml` ファイルをポイントするURI です。解析されていないエンティティの場合、エンティティの値は非-XML データ ファイルをポイントするURI です。

エンティティを定義する方法:

1. 「**Authentic | XML エンティティの定義**」をクリックします。エンティティの定義ダイアログが表示されます。



2. **名前**フィールドにエンティティ名を入力します。この値がエンティティ入力ヘルパーに表示される名前となります。
3. エンティティの型を型フィールドのドロップダウンリストから入力します。3つある種類から選択を行います。テキストが使用される **Internal** エンティティがXMLドキュメント自身内に保管されています。**PUBLIC** または **SYSTEM** では、エンティティのリソースが、XMLファイルの外に配置され、それぞれPUBLIC 識別子とSYSTEM 識別子を使用することで、リソースが特定されます。システム識別子ではリソースの場所を表すURI が記述されます。公開識別子はプロセッサーがリソースを識別するためのロケーションに影響されない識別子です。公開とシステム識別子の両方を指定すると、公開識別子はシステム識別子、システム識別子を使用して解決します。
4. PUBLIC がType として選択されている場合、PUBLIC フィールド内にリソースの公開識別子を入力します。Type にて Internal またはSYSTEM を選択した場合、PUBLIC フィールドが無効になります。
5. **Value/Path** フィールドでは、以下のどれかを入力することができます:

   - エンティティの種類がInternal の場合、エンティティの値となるテキスト文字列を入力してください。エントリに引用符を入力しないでください。入力する引用符はテキスト文字列の一部として扱われます。
   - SYSTEM がエンティティの型である場合、参照ボタンを使用してリソースのURI 、または、ローカルネットワーク上のリソースを入力します。リソースが参照可能エンティティである場合、参照されるリソースはXML ファイル(つまり **.xml** 拡張子を持ったファイル)である必要があります。他にも、GIF ファイルのようなバイナリファイルを使用することもできます。
   - エンティティの型がPUBLIC の場合、このフィールドにシステム識別子を追加で入力する必要があります。

6. NDATA エントリはプロセッサーにこのエンティティを解析せず適切なプロセッサーに送信されるように命令します。そのためNDATA フィールドは解析対象外エンティティに対してのみ使用されます。

ダイアログ機能

対応するボタンをクリックすることでエンティティの追加、挿入、そして削除を行うことができます。カラムのヘッダーをクリックすることで、エンティティをアルファベット順に並び替え。1回クリックすることで昇順、2回クリックで降順の並び替えとなります。

ダイアログボックスの大きさならびにカラムの幅を変更することができます。エンティティが1度 XML ドキュメント内で使用されると、そのエンティティはロックされ、エンティティの編集ダイアログにて編集することができなくなります。ロックされたエンティティは、最初の列内でロックシンボルにより表示されます。エンティティをロックすることにより、エンティティに対して XML ドキュメントが有効であることを保証することができます(エンティティが参照されており、定義されていない場合は、ドキュメントは無効になります)。

エンティティの複製はフラッグされます。

制約

- 他のエンティティ内に含まれているエンティティは、ダイアログ、Authentic View、XSLT 出力にて解決されず、そのようなエンティティのアンパサンドはエスケープされた形(つまり `&`)で表示されます。

- 外部エンティティは、そのエンティティがイメージである場合とENTITY または ENTITIES 型の属性値として入力された場合を除き Authentic View にて解決されません。このようなエンティティは、SPS から生成された XSLT にてドキュメントが処理されたときに解決されます。

24.9.8 マークアップの表示

Authentic XML ドキュメント内のマークアップをコントロールするオプションを使用するサブメニューがマークアップコマンドに搭載されています。これらのオプションは下で説明されています。

マークアップを隠すコマンドにより、Authentic View に表示されているマークアップシンボルを非表示にします。

小さなマークアップを表示コマンドにより、小さなマークアップシンボルを Authentic View にて表示します。

大きなマークアップを表示コマンドにより、大きなマークアップシンボルを Authentic View にて表示します。

混合マークアップを表示コマンドにより、異なるマークアップシンボルが Authentic View で表示されます。StyleVision Power Stylesheet では、デザイン時にドキュメントの各要素/属性に対して、大きなマークアップ、小さなマークアップ、またはマークアップ無しというオプションを指定することができます。混合マークアップのモードでは、カスタマイズされたマークアップが Authentic View に表示されます。

24.9.9 リッチエディット

リッチエディットコマンドへマウスを移動させることで、リッチエディットのマークアップに関するコマンドが含まれるサブメニューが表示されます(以下のスクリーンショットを参照)。このサブメニューに含まれるコマンドは、Authentic View にてリッチエディットコンポーネントの要素内にカーソルが配置された状態で選択可能になります。

リッチエディットメニューのテキストスタイルプロパティは、リッチエディットコマンドがクリックされた時に選択されたテキストへ適用されます。選択されたテキストにおけるフォント、フォントの太さ、装飾、大きさ、色、背景色、そして配置方法を Authentic View ユーザーは指定できます。

24.9.10 行の追加/挿入/複製/削除

行の追加コマンドにより、Authentic View の現在アクティブなテーブルに新たな行を追加します。

行の挿入コマンドにより、Authentic View の現在アクティブなテーブルに新たな行を挿入します。

行の複製コマンドにより、Authentic View の現在アクティブなテーブル行を複製します。

行の削除コマンドにより、Authentic View の現在アクティブなテーブル行を削除します。

24.9.11 上に/下に移動

行を上に移動コマンドにより、Authentic View における現在のテーブル行を、1行上に移動します。

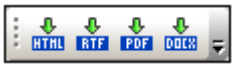
行を下に移動コマンドにより、Authentic View における現在のテーブル行を、1行下に移動します。

24.9.12 HTML、RTF、PDF、Word 2007+ ドキュメントの生成

以下にある4つのコマンドにより、PXF ファイル内部に収められた Authentic View XML ドキュメントから出力ドキュメントを生成することができます:

- **HTML ドキュメントの生成**
- **RTF ドキュメントの生成**
- **PDF ドキュメントの生成**
- **Word 2007+ ドキュメントの生成**

これらコマンドへは Portable XML Form (PXF) ツールバーからもアクセスすることができます(以下のスクリーンショットを参照)。



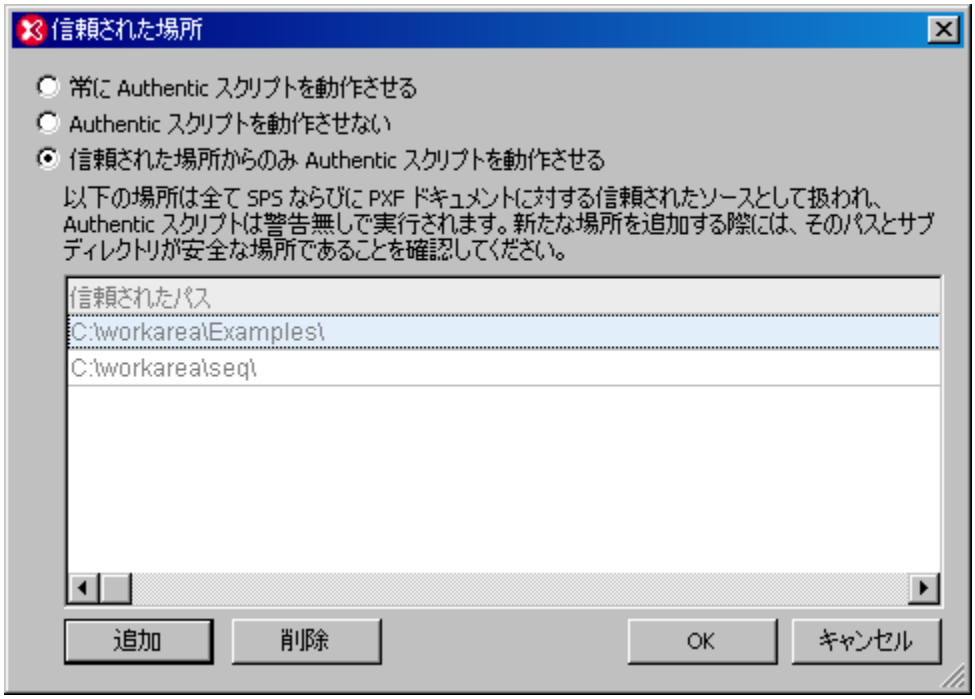
個々のコマンドまたはボタンをクリックすることで、HTML、RTF、PDF、または DocX 形式の出力がそれぞれ生成されます。

これらのボタンは PXF ファイルが Authentic View にて開かれた時に有効になります。PXF ファイル内にそれぞれの出力形式に対応した XSLT ファイルが含まれることで、個々のコマンドならびにボタンが有効になります。例えば、HTML ならびに RTF を出力する XSLT スタイルシートが PXF ファイル内部に含まれている場合、HTML ならびに RTF 出力を行うためのコマンドならびにツールバーボタンが有効になり、PDF と DocX (Word 2007+) 出力を行うコマンドは無効状態となります。

24.9.13 信頼された場所

信頼された場所コマンドを選択すると、(以下のスクリーンショットにある) 信頼された場所ダイアログが表示され、SPS 内にあるスクリプトのセキュリティー設定を指定することができます。スクリプトを含む SPS をベースにしている XML ファイルを Authentic ビューで開いた場合、このダイアログで指定された内容に従い、スクリプトの実行が許可されます(または許されません)。

(C) 2015-2021 Altova GmbH

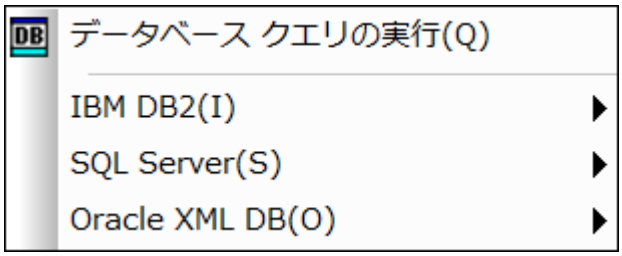


以下にある3つのオプションが指定可能です:

- Authentic ビューにてファイルが開かれた時に、Authentic スクリプトを常に動作させる。
- Authentic ビューにてファイルが開かれても、Authentic スクリプトを動作させない。
- 信頼された場所にある Authentic スクリプトだけを動作させる。信頼された場所(フォルダー)のリストが下部のペインに表示されます。**追加**ボタンを使用することで、フォルダーの参照とリストへの追加を行うことができます。リストからエントリーを削除するには、リストからエントリーを選択した後、**削除**ボタンをクリックしてください。

24.10 DB メニュー

DB メニューは(DB) オペレーションのためのメニューです。下のスクリーンショットに示されているように、下にリストされているメニューが含まれています。DB メニューのサブメニュー内のコマンドの説明は、このセクションのサブセクションを参照してください。

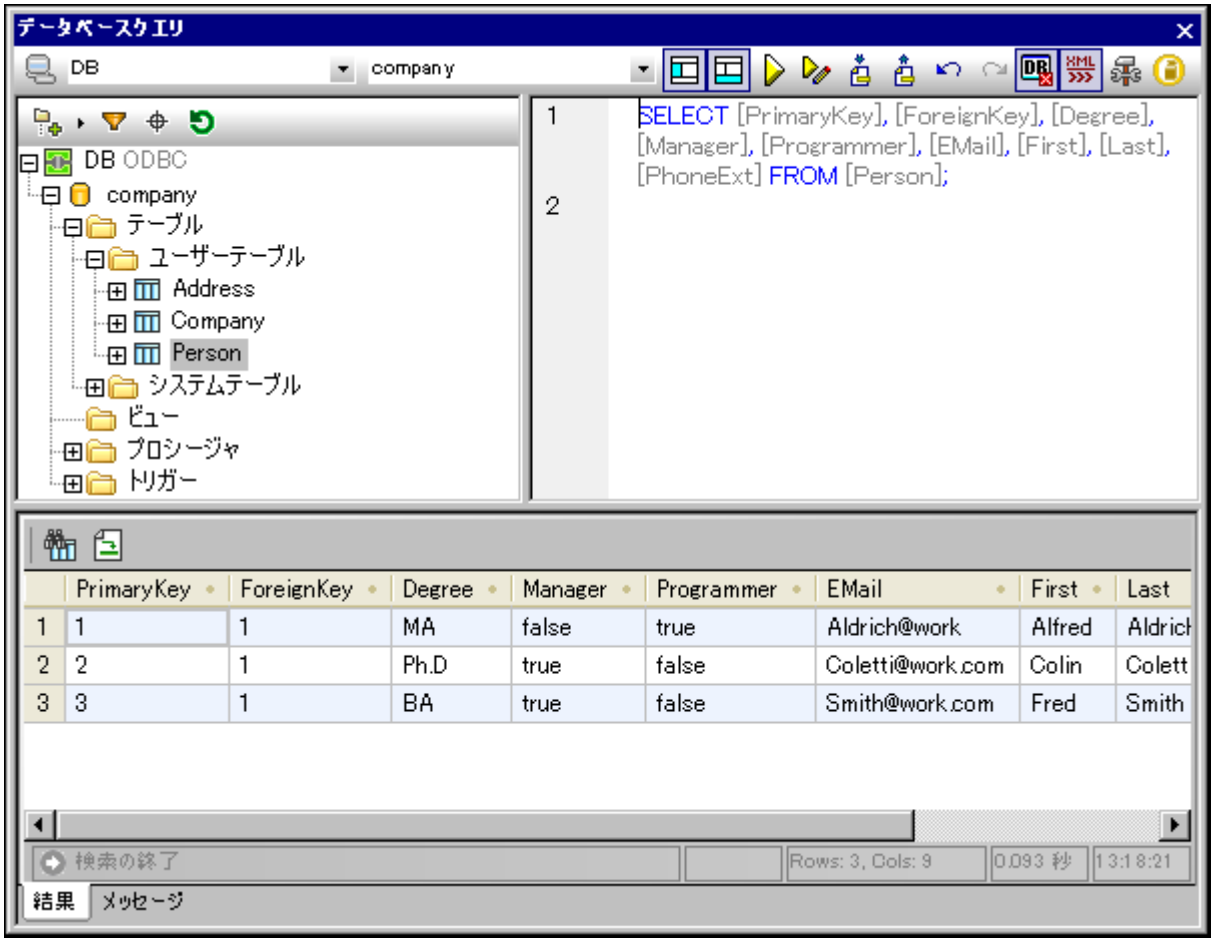


- クエリデータベース 912 :各種データベースの問い合わせを行うことができます。
- IBM DB2 925 :DB2 に特化した機能がサポートされます。
- SQL サーバー 931 :SQL Server データベースの管理を行うためのコマンドが収められます。.

このセクションで記述されている全てのコマンドを実行するには、データベースへの接続を行う必要があります。詳細に関してはデータベースへの接続 573 を参照してください。

24.10.1 データベースをクエリする

データベースクエリの実行コマンドでは、データベースクエリウィンドウが開かれます(以下のスクリーンショットを参照)。クエリウィンドウが開かれると、「DB | **データベースクエリの実行**」コマンドまたは、データベースクエリツールバーアイコン により、表示を切り替えることができます。



データベースクエリウィンドウの概要

データベースクエリウィンドウは、以下のパーツにより構成されています:

- 左上にある ブラウザーペイン 917 により、接続情報ならびにデータベーステーブルが表示されます。
- 右上にある クエリペイン 920 によりクエリが入力されます。
- タブ化されている 結果/メッセージペイン 923 が下部に表示されます。結果ペインにはクエリの結果が結果グリッドと呼ばれるテーブルに表示されます。メッセージペインでは、警告、エラーを含むクエリの実行に関する情報が表示されます。

データベースクエリウィンドウの上部にはツールバーが用意されています。ここでは以下にある2つのツールバーアイコンに注目してください。他のツールバーアイコンに関する情報については、クエリペイン: 詳細と機能 920 を参照ください。

ブラウザーペインの表示を切り替えます。

結果/メッセージペインの表示を切り替えます。

クエリデータベースメカニズムの概要

クエリデータベースは、以下のように動作します。詳細に関しては、このセクション以下にあるサブセクションを参照ください。

1. データベースクエリウィンドウにてデータベースへの接続が確立[914] されます。サポートされているデータベースには、MS Access 2000 ならびに2003、Microsoft SQL Server、Oracle、MySQL、Sybase、そしてIBM DB2 があります。
2. 接続されたデータベースは、ブラウザーペイン[917]にて表示され、最適な表示を行うための設定を施すこともできます。
3. 選択されているデータベースに対して適切な構文で記述された query[922] が、Query ペイン[922], に入力され、クエリが実行されます。
4. クエリの結果[923] は、様々なフィルターにより閲覧、編集、そしてDB への保存することができます。

24.10.1.1 データソース

データベースに対してクエリを実行するには、まず目的のデータベースへ接続する必要があります。このセクションでは、以下の方法について記述します:

- データベースへの接続を行い、
- 複数ある既存の接続から、目的のデータソースとルートオブジェクトを選択

データベースへの接続

セッションにて初めて(または既存のデータベース接続がない状態で) **データベースクエリの実行**コマンドをクリックすると、クイック接続ダイアログが表示され、データベースへの接続を行うことができます。作成した接続をそれ以降も使用する場合、データベースクエリウィンドウにあるクイック接続アイコン をクリックしてください。

クイック接続ダイアログからデータベースへ接続する方法については、データソースへの接続 573 セクションを参照ください。

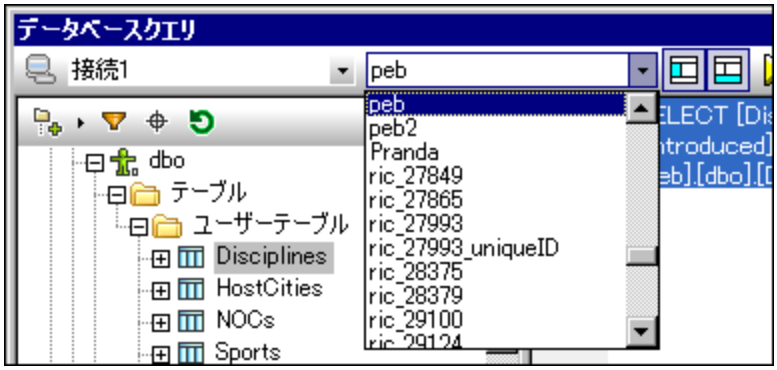
次のデータベースはサポートされています。各データベースの使用できるルート オブジェクトもリストされています。Altova は他の ODBC/ADO データベースへのサポートの向上のための努力をしていますが、以下にリストされたデータベースのみで接続とデータ処理のテストに成功しています。使用中の Altova アプリケーションが、64 ビット バージョンの場合、接続されているデータベース特定の 64 ビット データベース ドライバーにアクセスしてください。Altova アプリケーションは IBM iSeries データベースのロジカル ファイルをサポートし、ロジカル ファイルを表示します。

データベース	メモ
Firebird 2.5.4, 3.0	
IBM DB2 8.x, 9.1, 9.5, 9.7, 10.1, 10.5	
IBM Db2 for i 6.1, 7.1, 7.2, 7.3, 7.4	ロジカルファイルはサポートされ、ビューとして表示されています。
IBM Informix 11.70, 12.10, 14.10	
MariaDB 10.2, 10.3, 10.4, 10.5	
Microsoft Access 2003, 2007, 2010, 2013, 2016, 2019	ドキュメントの作成時 (2019 年9 月初旬) Access 2019 のために使用可能な Microsoft Access Runtime が存在しませんでした。Microsoft Access 2016 Runtime がインストールされており、データベースが「大量」のデータ型を使用しない場合のみ Altova 製品から Access 2019 にデータベース接続することができます。

データベース	メモ
Microsoft Azure SQL Database	SQL Server 2016 codebase
Microsoft SQL Server 2005, 2008, 2012, 2014, 2016, 2017, 2019 Microsoft SQL Server on Linux	
MySQL 5.0, 5.1, 5.5, 5.6, 5.7, 8.0	
Oracle 9i, 10g, 11g, 12c, 18, 19	
PostgreSQL 8.0, 8.1, 8.2, 8.3, 9.0.10, 9.1.6, 9.2.1, 9.4, 9.5, 9.6, 10, 11, 12.1	PostgreSQL 接続は、ODBC または JDBC などのインターフェイス(ドライバ)によりネイティブドライバーベースの接続でサポートされています。ネイティブな接続はドライバを必要としません。
Progress OpenEdge 11.6	
SQLite 3.33.0	SQLite 接続は、ネイティブとしてサポートされます。SQLite データベースファイルへの直接の接続です。 別途のドライバーは必要ありません。Authentic ビューでは、SQLite からのデータを編集することはできません。Authentic ビューで、SQLite からのデータを保存する場合は、既知の制限についてのメッセージが通知されます。
Sybase ASE 15, 16	
Teradata 16	

目的のデータソースを選択

既存の接続ならびにルートオブジェクトが、データベースクエリウィンドウのツールバーにある2つのコンボボックスにてそれぞれ表示されます(以下のスクリーンショットを参照)。



上のスクリーンショットでは、StyleVision DB という名前のデータベースが選択されています。このデータベースにて利用することのできるルートオブジェクトのうち、ALTOVA_USER が選択されています。データベースならびにルートオブジェクトはブラウザーペインにて表示されます。

24.10.1.2 ブラウザーペイン：DB オブジェクトの閲覧

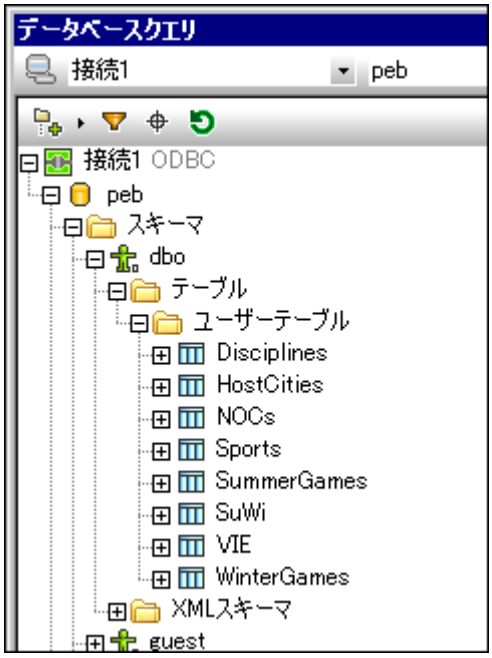
ブラウザーペインでは、選択されたデータベースにあるオブジェクトの概要が表示されます。この概要には、どのカラムが主キーや外部キーかといった、データベース制約情報も表示されます。IBM DB2 バージョン 9 データベースでは、登録された XML スキーマも別のフォルダーにてブラウザービューに表示されます。

このセクションでは以下のことがらに関する記述を行います:

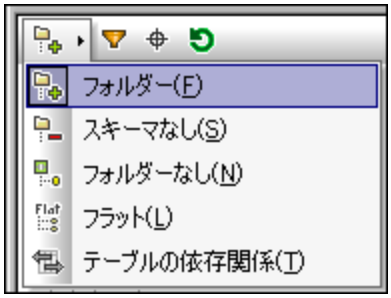
- ブラウザーペインにて利用可能な レイアウト 917
- データベースオブジェクトのフィルタリングの方法 918
- データベースオブジェクトの検索の方法 919

ブラウザーペインレイアウト

デフォルトのフォルダーレイアウトでは、データベースオブジェクトが階層構造で表示されます。選択されたオブジェクトにより、右クリックにより表示されるコンテキストメニューのオプションが変化します。



ブラウザーのレイアウトを選択するには、ブラウザーペインのツールバーにあるレイアウトアイコンをクリックし、ドロップダウンリストからレイアウトを選択します（以下のスクリーンショットを参照）。選択されたレイアウトによりアイコンが変化することに注意してください。



利用可能なレイアウトは以下のとおりです:

- *フォルダー*：データベースオブジェクトを、オブジェクトの種類ごとにフォルダー形式の階層構造で表示します。デフォルトの設定です。
- *スキーマなし*：フォルダーレイアウトに似ていますが、データベーススキーマフォルダーは表示されません。そのため、テーブルはデータベーススキーマによりカテゴライズされません。
- *フォルダーなし*：フォルダーを使用すること無く、データベースオブジェクトを階層構造で表示します。
- *フラット*：データベースオブジェクトを、型ごとに最初の階層構造レベルにおいて割り振ります。例えば、個々のカラムが対応するテーブルに含められるのではなく、全てのカラムが特定のカラムフォルダーに収められて表示されます。
- *テーブルの依存関係*：他のテーブルとのリレーションシップに従い、テーブルをカテゴライズします。外部キー付きのテーブル、外部キーにより参照されているテーブル、そして他のテーブルに対してリレーションシップを持たないテーブルがカテゴリーとして表示されます。

テーブルをユーザーテーブルならびにシステムテーブルで並べ替えるには、スキーマなし、またはフラットレイアウトへの切り替えを行い、テーブルフォルダーを右クリックして、**ユーザーとシステムテーブルの並べ替え**を選択します。ユーザーテーブルならびにシステムテーブル内のテーブルがアルファベット順に並べ替えられます。

データベースオブジェクトのフィルタリング

ブラウザーペインでは、（フォルダーなしと、テーブルの依存関係レイアウトを除き）スキーマ、テーブル、そしてビューを、名前や名前の一部によりフィルタリングすることができます。オブジェクトのフィルタリングは文字の入力により行われます。フィルタリングにおいて、大文字と小文字は区別されません。

以下の操作により、ブラウザーペインにてオブジェクトのフィルタリングを行うことができます:

1. ブラウザーペインのツールバーにあるフォルダーコンテンツのフィルターアイコンをクリックします。フィルターアイコンが、テーブルならびにビューフォルダーの隣に表示されます（以下のスクリーンショットを参照）。

2. 新たに表示される入力フィールドにて、フィルタリングを行う文字列を入力します（以下のスクリーンショットでは、テーブルフォルダーに対して Su というフィルタリングが行われています）。入力しながらも、フィルタリングは行われます。

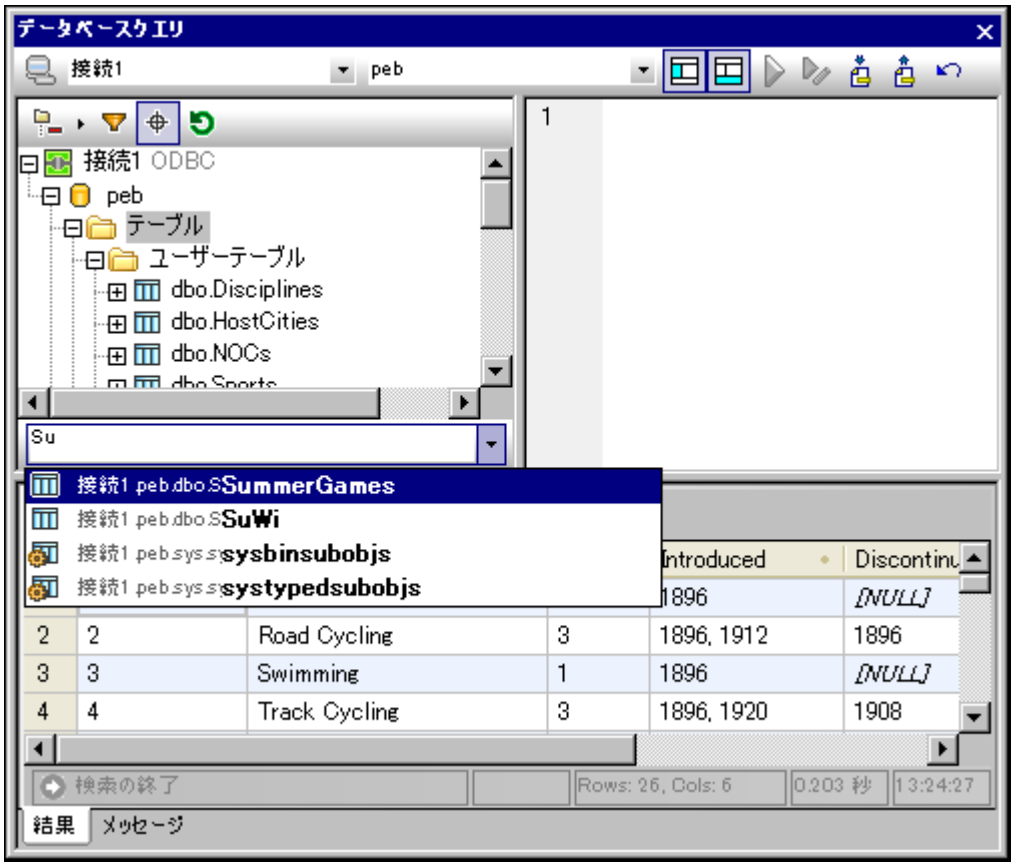
3. 新たに表示される入力フィールドにて、フィルタリングを行う文字列を入力します（以下のスクリーンショットでは、テーブルフォルダーに対して Su というフィルタリングが行われています）。入力しながらも、フィルタリングは行われます。

(C) 2015-2021 Altova GmbH

データベースオブジェクトの検索

以下にあるように、ブラウザーペインのオブジェクトロケーターを使用するで、特定の名前を持ったデータベースアイテムを検索することができます。

1. ブラウザーペインのツールバーにて、オブジェクトロケーターアイコンをクリックします。ブラウザーペインの下部にドロップダウンリストが表示されます。
2. リストの入力フィールドに検索文字列を入力します。以下のスクリーンショットでは、例として Su という文字列が入力されています。ドロップダウンリストの矢印をクリックすることで、その検索文字列を含む全てのオブジェクトが表示されます。



3. リスト内のオブジェクトをクリックして、ブラウザーペインにて確認します。

24.10.1.3 クエリペイン：説明と機能

クエリペインは、選択されたデータベースに対してクエリを実行するためのインテリジェントな SQL エディターです。クエリを入力した後は、データベースクエリウィンドウの実行コマンドをクリックすることでクエリが実行され、その結果と実行メッセージが結果/メッセージペイン 923 にて表示されます。クエリの操作方法については、クエリペイン：クエリの使用方法 922 を参照ください。このセクションでは、以下に示されるクエリペインの主要な機能を記述します。

- データベースクエリツールバーにある SQL エディターアイコン
- SQL エディターオプション
- SQL ステートメントの自動補完
- SQL スクリプトの領域定義
- SQL スクリプト内へのコメント挿入
- ブックマークの使用

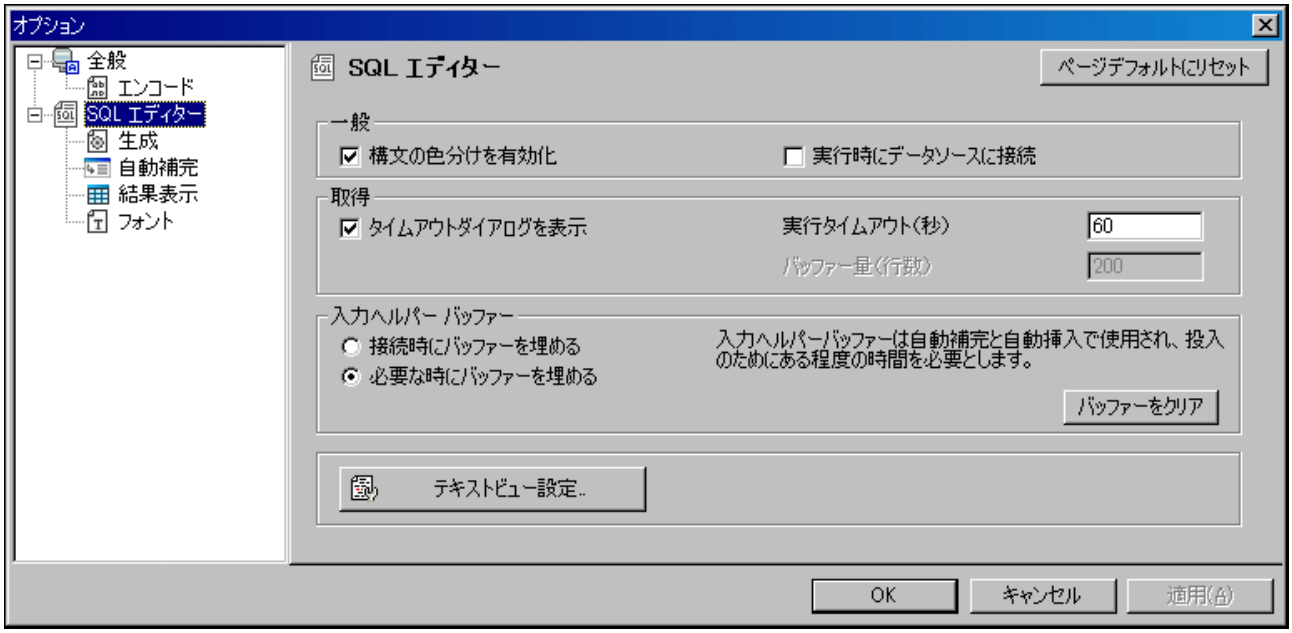
データベースクエリツールバーの SQL エディターアイコン

データベースクエリウィンドウのツールバーでは以下のアイコンが用意されており、SQL エディターを使用する際に利用することができます：

	実行	現在選択されている SQL ステートメントを実行します。スクリプトに複数のステートメントが含まれており、そのうちどれも選択されていない場合、全てのステートメントが実行されます。
	データ編集の実行	実行コマンドと同じですが、結果タブに表示される結果を編集することができます。
	SQL ファイルのインポート	SQL ファイルを SQL エディターにて開きます。
	SQL ファイルのエクスポート	SQL クエリを SQL ファイルに保存します。
	元に戻す	SQL エディターでなされた変更を回数に制限なく元に戻します。
	やり直し	SQL エディターでなされた「元に戻す」コマンドからやり直しを行います。
	XML を開く際に DB クエリを隠す	XML ドキュメントが開かれたときに、DB クエリウィンドウを隠すかどうか選択します。
	XML 保存時に自動コミット	編集された XML ドキュメントを XMLSpy にて保存する際に、このボタンが有効になっていれば、DB へのコミットが自動的に行われます。そうでない場合、結果ペインから明示的にコミットを行う必要があります。
	オプション	SQL エディターのオプションダイアログを開きます。
	SQL スクリプトを DatabaseSpy で開く	Altova の DatabaseSpy にて SQL スクリプトを開きます。

オプション

データベースクエリツールバーの**オプション**アイコンをクリックすることで、オプションダイアログが表示されます（以下のスクリーンショットを参照）。左側にあるペインから設定のページを選択し、そのページのオプションを指定することができます。**ページデフォルトにリセット**ボタンをクリックすることで、そのページのオプションを、デフォルトの設定にリセットすることができます。

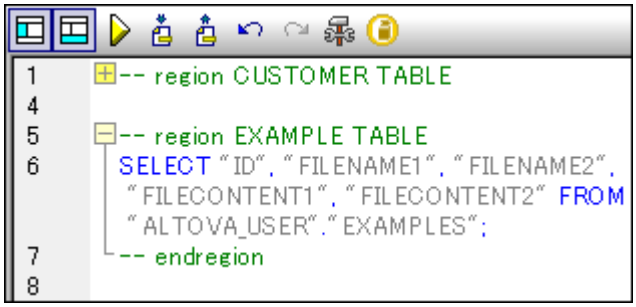


重要な設定を以下に記します:

- 全般 / エンコード :新たな SQL ファイルや、エンコーディングを特定できない既存 SQL ファイルのエンコーディングならびにバイトオーダーマーク(BOM) をセットするオプションです。既存 SQL ファイルのエンコーディングを特定できた場合、エンコーディングを変更すること無く、開かれたファイルは保存されます。
- SQL エディター :構文による色付けや実行時のデータソース接続などを有効にすることができます。クエリの実行にはタイムアウトを設定することができ、指定された値を超えて時間が経過すると、タイムアウトの変更を促すダイアログが表示されます。入力ヘルパーのバッファーは、データソースへの接続時、またはそれが最初に求められた際に埋められるよう指定することができます。テキストビュー設定ボタンによりXMLSpy のテキストビューオプションウィンドウが開きます。.
- SQL エディター / SQL 生成 :オブジェクトをブラウザーペインからクエリペインへドラッグすると、アプリケーションによりSQL ステートメントが生成されます。SQL ステートメント生成に関するオプションは、SQL 生成タブにて指定することができます。データベースペインからデータベースの種類を選択することで、各種データベースに対して個別のオプションを指定することができます。全てのデータベースに適用チェックボックスにチェックを入れることで、現在選択されているオプションが、全種類のデータベースに対して適用されます。オプションには、セミコロンをステートメントに追加したり、識別子をエスケープ文字により囲むなどがあります。エディターによりセミコロンを追加 チェックボックスが有効になっている場合、SQL ステートメントがSQL エディターにて生成された際に、セミコロンが追加されます。JDBC 接続によりOracle データベース、IBM iSeries、そしてDB2 データベースへ接続し、データの編集を行うには、このチェックボックスのチェックを外す必要があります。
- SQL エディター / 結果ビュー :結果タブのオプションを指定します。
- SQL エディター / フォント :テキスト エディターならびに結果ビューのフォントを指定します。

SQL スクリプトの領域を定義

領域とは、SQL スクリプトにて選択され、1つのまとまりとして宣言された箇所のことです。領域により、縮退や展開、スクリプトの一部だけ非表示にすることが可能になります。領域を他の領域の中にネストすることもできます。領域の開始位置と終了位置は、-- region ならびに -- endregion コメントにより、それぞれ定義されます。オプションとして、-- region の後ろに文字列を入力することで、領域に名前を付けることができます(以下のスクリーンショットを参照)。



領域を挿入するには、領域に含めるステートメントを選択し、右クリックにて表示されるコンテキストメニューから、**領域の挿入**を選択します。展開/縮退可能な領域が作成されます。必要な場合は、領域に名前を追加してください。上のスクリーンショットで表示されている行番号にも注目してください。領域を削除するには、-- region ならびに-- endregion を削除してください。

SQL スクリプトにコメントを挿入

SQL スクリプトに表示されているテキストはコメントアウトすることができます。これらコメントはスクリプトの実行時にスキップされます。

- ブロックのコメントアウトを行うには、ブロックを選択して、右クリックで表示されるコンテキストメニューから、ブロックコメントを挿入/削除を選択します。ブロックコメントを削除するには、コメントを選択し、右クリックからブロックコメントを挿入/削除を選択します。
- 行や、行の一部をコメントアウトするには、行コメントを挿入する場所にカーソルを移動し、右クリックから行コメントを挿入/削除を選択します。行コメントを削除するには、コメントを選択し、右クリックから行コメントを挿入/削除を選択します。

ブックマークの使用

特定の行にブックマークを挿入し、ドキュメント内のブックマークによりナビゲーションを行うことができます。ブックマークを挿入するには、ブックマークする行にカーソルを移動させ、右クリックにより表示されるコンテキストメニューから、**ブックマークの挿入/削除**を選択します。次や前のブックマークに移動するには、右クリックから**次のブックマークへ移動**または**前のブックマークへ移動**を選択します。ブックマークを削除するには、削除するブックマークの行にカーソルを移動させ、右クリックから**ブックマークの挿入/削除**を選択します。全てのブックマークを削除するには、右クリックから**全てのブックマークを削除**を選択します。

24.10.1.4 クエリペイン：クエリの実行

データベースへの接続を行うと、SQL エディターにて SQL スクリプトを入力、実行することができます。このセクションでは以下のことがらについて記述します:

- SQL エディターにて SQL スクリプトを入力する方法
- データベースクエリウィンドウにて SQL スクリプトを実行する方法

このセクションでは以下のアイコンが使用されます:

	クエリの実行	現在選択されている SQL ステートメントを実行します。スクリプトに複数のステートメントが含まれており、そのうちどれも選択されていない場合、それら全てが実行されます。
	データ編集の実行	実行コマンドと同じですが、(結果タブに表示される) 結果を編集することができます。

SQL ファイルのインポート SQL ファイルを SQL エディターにて開きます。

SQL エディターにてSQL ステートメントならびにスクリプトを作成

以下のGUI 操作からSQL ステートメントやスクリプトを作成することができます:

- ドラッグアンドドロップ：ブラウザーペインからSQL エディターへオブジェクトをドラッグします。そのオブジェクトをデータベースから問い合わせるためのSQL ステートメントが生成されます。
- コンテキストメニュー：ブラウザーペインにあるオブジェクトをクリックし、「新規 SQL エディターの表示 | SELECT」を選択します。
- マニュアル入力：SQL ステートメントをSQL エディターに直接入力します。自動補完機能を使って編集を行うことができます。
- SQL スクリプトのインポート：データベースクエリウィンドウのツールバーにあるSQL ファイルのインポートアイコンをクリックします。

SQL ステートメントの実行

SQL エディターのSQL スクリプトに2つ以上のSQL ステートメントが含まれている場合、実行するステートメントを選択して、データベースクエリウィンドウのツールバーにある**実行**アイコンまたは**データ編集の実行**アイコンをクリックします。SQL スクリプトが選択されていない場合は、スクリプト内にある全てのステートメントが実行されます。データベースのデータが取得され、結果タブ923. のグリッドに表示されます。**データ編集の実行**が選択された場合、結果タブのグリッドに表示されたデータを編集することができます923。実行に関するメッセージは、メッセージタブ923に表示されます。

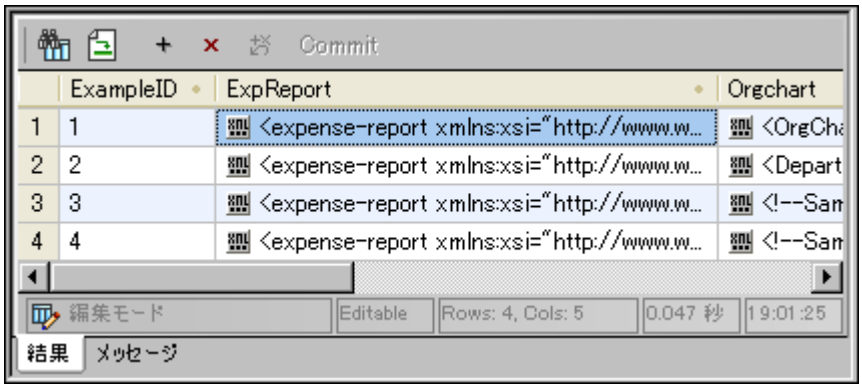
24.10.1.5 結果とメッセージ

結果/メッセージペインには2つのタブが含まれています:

- 結果タブ923 にはクエリにより取得されたデータが表示されます。
- メッセージタブ925 にはクエリの実行に関するメッセージが表示されます。

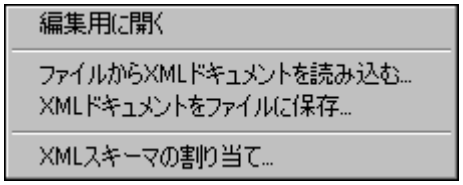
結果タブ

クエリにより取得されたデータは、グリッド形式で結果タブに表示されます(*以下のスクリーンショットを参照*)。**クエリの実行** コマンドによりクエリの結果が得られた場合、結果タブ内のXML ドキュメントはXML アイコンにより表示されます(*下のスクリーンショット*)。 **データ編集の実行** コマンドが使用された場合 XML ドキュメントはXML アイコンにより表示されます。



次の操作は、コンテキスト結果タブ内の適切な場所を右クリックするとポップアップする、コンテキストメニューを使用して、結果タブで実行することができます。

- *列の並べ替え:* 記録が並べ替えられる列を右クリックし、[**並べ替え | 昇順/降順/既定値に復元**] を選択します。。
- *クリップボードにコピー:* この手順には2 つのステップがあります: (i) データ範囲の選択、(ii) 選択範囲のコピー。データは複数の方法で選択することができます。: (i) 列のヘッダーまたは行番号をクリックしてれつまたは行を選択、(ii) 個別のセルを選択 (**Shift** キーおよびまたは **Ctrl** キーを使用して複数のセルを選択します); (iii) セルを右クリックして、[**選択 | 行/列/すべて**] を選択します。。選択が決定されると、右クリックして [**選択されたセルのコピー**] を選択します。この操作により、選択対象がクリップボードにコピーされ、他のアプリケーションの処理に渡すことができるようになります。
- *新規の行の追加:* 編集を行えるようクエリが実行された場合、結果ペインを右クリックすることで、**行の追加** コマンドへアクセスすることができます。
- *行の削除:* 新しい行の追加 : 編集を行えるようクエリが実行された場合、結果ペインを右クリックすることで、**行の削除**コマンドへアクセスすることができます。
- *レコードの編集 :* 編集を行えるようクエリが実行された場合、各フィールドを編集することができます。変更をコミットするには、結果ペインのツールバーにある **コミット** ボタンをクリックしてください。
- *XML レコードの編集:* この機能はIBM DB2, SQLServer、PostgreSQL (8.3 または以降)、および Oracle (9 または以降) データベースと主キーを持つDB テーブルのためにサポートされています。データ編集の実行コマンドにより編集可能な XML フィールドが取得された場合、結果グリッドにて編集可能 XML アイコン をクリックすることで、XML 編集メニューが表示されます(以下のスクリーンショットを参照)。ツリーから XML フィールドを右クリックして、**データ編集**を選択することでも XML フィールドを開くことができます。



編集用に開くコマンドにより、XML ドキュメントが XMLSpy ウィンドウで開かれ、編集可能アイコン内のドットが赤色に変化 します。ドキュメントが開かれたときに、データベースクエリツールバーにある XML の変更を自動コミットアイコン が有効になっている場合、このドキュメントが保存されるとともに、XML に対して行われた変更が自動的にデータベースへコミットされます。アイコンが有効になっていない場合、XML に対して保存された変更を、結果ペインにあるコミットボタンによりコミットする必要があります。XML ドキュメントとデータベースクエリウィンドウを切り替えるには、「**DB | データベースクエリの実行**」コマンドをクリックする必要があります。**ファイルから XML ファイルを読み込む** コマンドにより、外部の XML ドキュメントをデータベースの選択されたフィールドへロードすることができます。**XML ドキュメントをファイルに保存** コマンドでは、選択されたデータベースフィールドの XML ドキュメントを、ユーザーが選択する場所へ保存することができます。XML スキーマの割り当てコマンドを選択すると、XML スキーマの選択 ダイアログ 929 が表示され、XML ドキュメントへ割り当てる XML スキーマを選択することができます。この割り当て情報はデータベースへ保存されます。XML スキーマの割り当てに関する詳細は、「IBM DB2 | XML スキーマの割り当て 929 のセクションを参照ください。
- *NULL のセット、デフォルトのセット、このセルの変更を取り消す :* 編集を行えるようクエリが実行された場合、セルを右クリックすることで、そのセルに対して NULL 値や、(デフォルト値が定義されている場合) セルのデフォルト値をセットすることができます。セルに対して行われた変更は、**このセルの変更を取り消す**コマンドにより、表示されている値を DB に収められている値に戻すことができます。

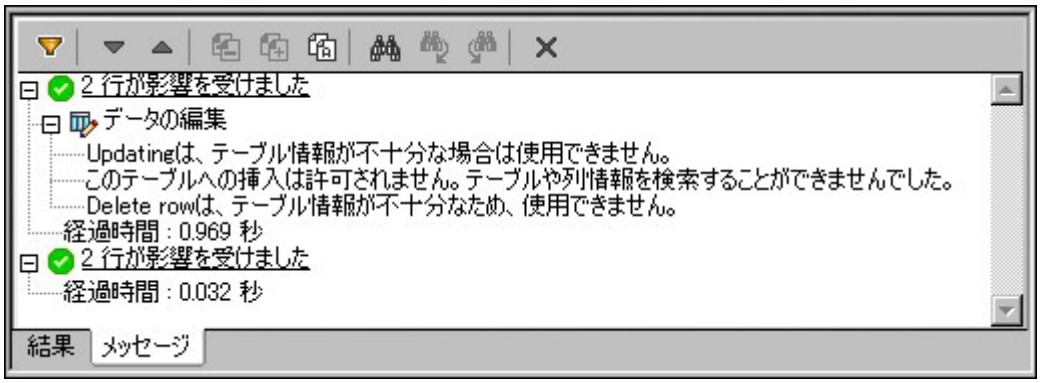
フォーミュラタブには以下のツールバーアイコンがあります:

	ステートメントへ移動	表示されている結果を取得するためのSQL エディターにてステートメントがハイライトされます。
	検索	結果ペインにあるテキストを検索します。XML ドキュメントのコンテンツも検索されます。
	新しい行の追加	新たな行を結果グリッドへ追加します。
	行の削除	現在選択されている行を結果グリッドから削除します。

	結果グリッドでの変更を全て元に戻す	結果グリッドでなされた全ての変更を元に戻します。
Commit	**コミット**	結果グリッドでなされた全ての変更をデータベースへコミットします。

メッセージタブ

メッセージタブにはそれまでに実行されたSQL ステートメントの情報がエラーや警告メッセージとともに表示されます。



メッセージタブのツールバーにはビューのカスタマイズやナビゲーション、クリップボードへのメッセージコピーを行うためのアイコンが含まれています。[フィルター] アイコンにより、特定の種類のメッセージの表示の切り替えをオンまたはオフします。「次」ならびに「前」アイコンにより、選択箇所を移動することができます。表示されているメッセージは、子コンポーネントを含めて(または含めないで)クリップボードにコピーすることができ、その後ドキュメントへの貼付けを行うことができます。[検索] 機能により、検索用語を指定し、この用語をリスト内上下に検索することができます。更に、クリアアイコンを使用すると、メッセージペインに表示されているコンテンツをクリアすることができます。

メモ ツールバーアイコンコマンドは、コンテキスト メニューコマンドとしても使用することができます。

24.10.2 IBM DB2

IBM DB2 メニューアイテムには、(i) IBM DB2 データベースにおけるスキーマの登録や登録取り消し(XML スキーマの管理 925)ならびに(ii) XML ファイル検証で使用されるスキーマの割り当て(XML スキーマの割り当て 929)を行うためのコマンドが含まれています。

これらのコマンドを実行するには、IBM DB2 データベースへの接続を行う必要があります。データベースへの接続方法については、IBM DB2 (ODBC) への接続 609 セクションを参照ください。このセクションでは、IBM DB2 データベース内にてスキーマを管理する方法と、XML スキーマを DB XML ファイルに割り当てる方法について記述します。

メモ: データベースクエリウィンドウ 912 の結果グリッドには、IBM DB2データベース内にある XML ファイルを処理するための重要な機能が収められています。この機能を使うことで、XML ファイルを開き、DB XML ファイルへロード、DB XML ファイルを外部的に保存したり、XML スキーマを DB XML ファイルへ割り当てるといった操作を行うことができます。

24.10.2.1 XML スキーマの管理

XML スキーマの管理機能により、個々のデータベーススキーマから、IBM DB2 データベースへスキーマを追加したり、削除することができます。以下の操作によりスキーマの管理を行うことができます:

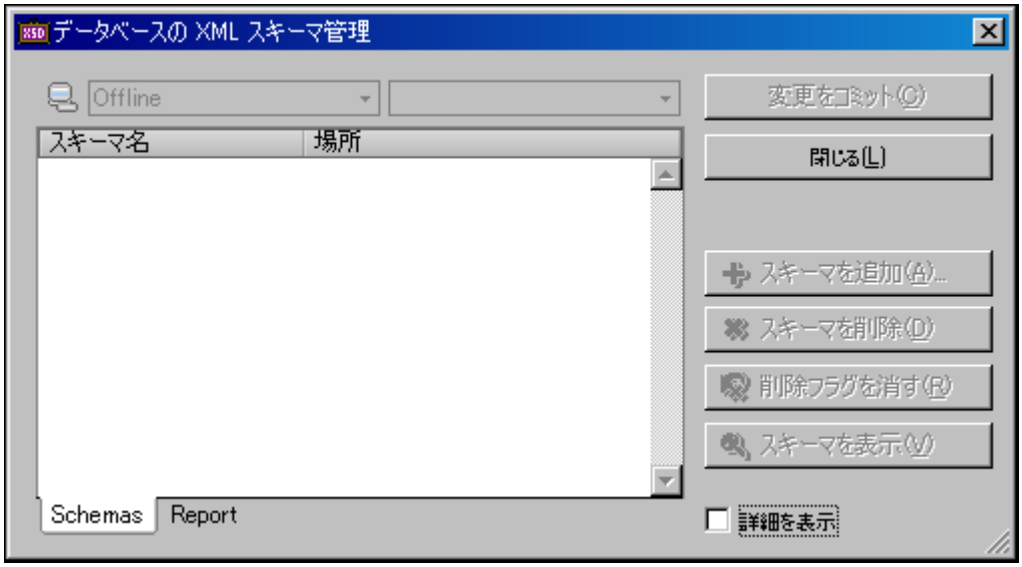
- IBM DB2 データベースへの接続

- XML スキーマを追加または削除するためのデータベーススキーマを選択
- スキーマの管理操作を行う

これらステップに関する詳細を以下に記します。

IBM DB2 データベースへの接続

「XML スキーマの管理」コマンドをクリックすることで、データベースの XML スキーマ管理ダイアログが表示されます（*以下のスクリーンショットを参照*）。

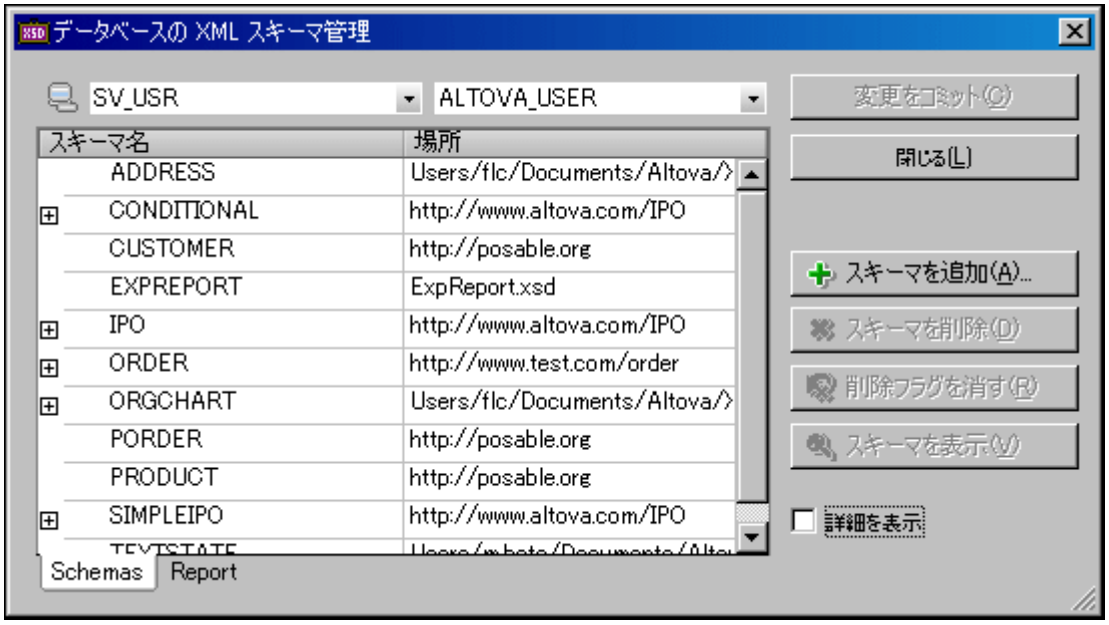


目的のデータベースに対して接続がなされていない場合、まずは接続を行う必要があります。既に接続が確立されている場合、その情報がデータベースコンボボックスに表示されます。接続処理を開始するには、ダイアログのクイック接続アイコン をクリックします。クイック接続ダイアログの使用方法については、データベースへ接続 573 セクションを参照ください。

XML スキーマのリストを表示

IBM DB2 データベースへの接続が確立されると、データベースがコンボボックスの中に表示されます（*以下のスクリーンショットを参照*）。複数のデータベースへの接続が行われている場合、このコンボボックスから目的のデータベースを選択することができます。以下のスクリーンショットでは、SV_USR データベースが選択されています。

(C) 2015-2021 Altova GmbH



右にあるコンボボックスには、現在選択されているIBM DB2 データベースにあるデータベーススキーマのリストが表示されます。データベーススキーマがこのコンボボックスで選択されると、そのデータベーススキーマに登録されている全てのXML スキーマがメインペインにて表示されます。上のスクリーンショットでは、`Altova_User` データベーススキーマに登録された全てのXML スキーマが、配置されている場所とともに表示されています。詳細を表示チェックボックスにチェックを入れることで、更に詳細な情報がメインペインに表示されます。

XML スキーマの管理

選択されたデータベーススキーマに対して登録されているスキーマがメインペインにて表示されます。XML スキーマのリストが表示されると、スキーマをリストに加えたり、リストからスキーマを削除することができます。

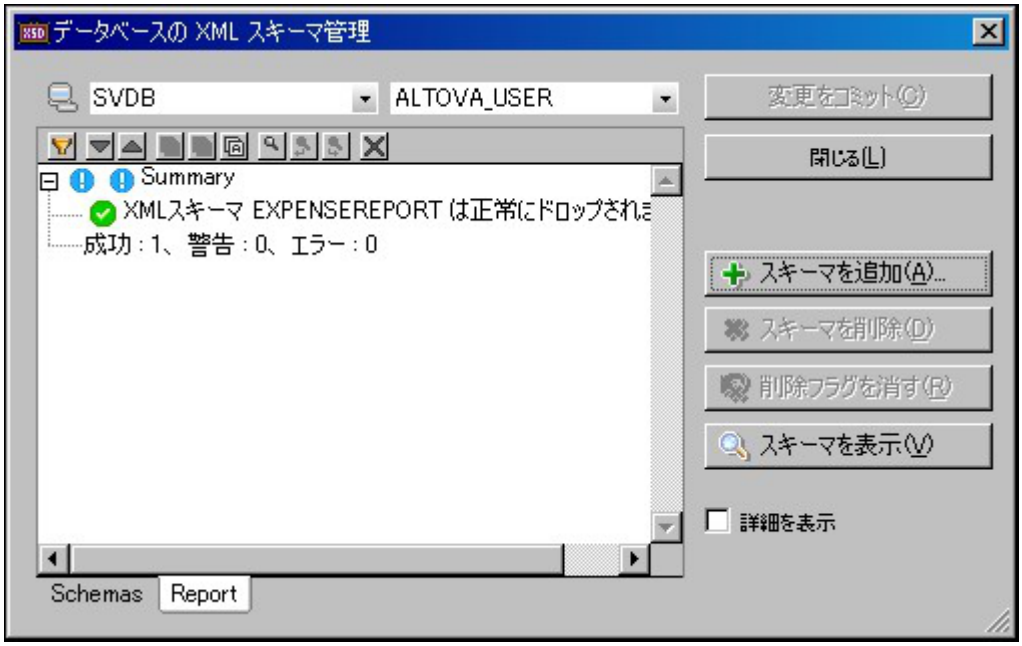
スキーマを追加するには、「**追加**」ボタンをクリックし、目的のスキーマファイルを指定します。選択されたスキーマファイルがリストに表示されます。「**変更をコミット**」ボタンをクリックすることで、新たに追加されたスキーマがデータベーススキーマに登録されます。

スキーマを削除するには、メインペインにてスキーマを選択し、「**スキーマを削除**」ボタンをクリックします。削除フラグがそのスキーマに割当てられ、変更が次にコミットされる際に、そのスキーマが削除されることを示します。フラグが立てられたスキーマを選択し、「**削除フラグを消す**」ボタンをクリックすることで、削除フラグを取り消すことができます。「**変更をコミット**」ボタンをクリックすると、削除フラグが立てられている全てのスキーマに対して、データベーススキーマの登録解除が行われます。

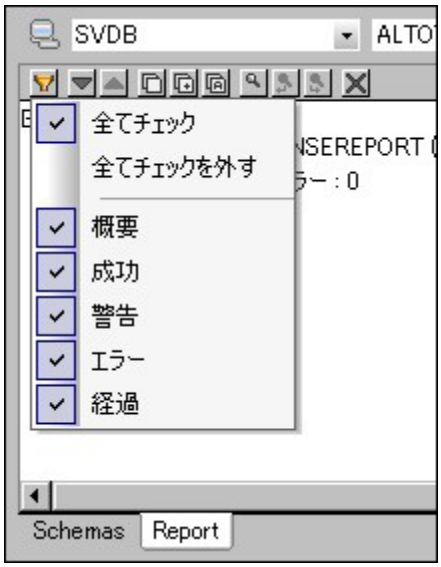
スキーマを表示ボタンをクリックすることで、スキーマがXMLSpy で開かれます。XML スキーマ管理ダイアログを閉じるには、「**閉じる**」ボタンをクリックしてください。

レポート

「**変更をコミット**」ボタンをクリックすると、加えられた変更に従うかたちでデータベースが変更されます。コミット処理のレポートは、レポートペインに表示され、処理が正常に行われたかの確認や、エラーが発生した際のデバッグに役立てることができます(*以下のスクリーンショットを参照*)。新たなレポートメッセージは、それまでのメッセージに続くかたちで表示されます。新たなレポートメッセージは、それまでのメッセージに続くかたちで表示されます。



レポートペインにあるツールバーには、表示のカスタマイズやナビゲーション、クリップボードにレポートをコピーしたり、テキストの検索やクリアを行うためのアイコンが含まれています(*以下のスクリーンショットを参照*)。



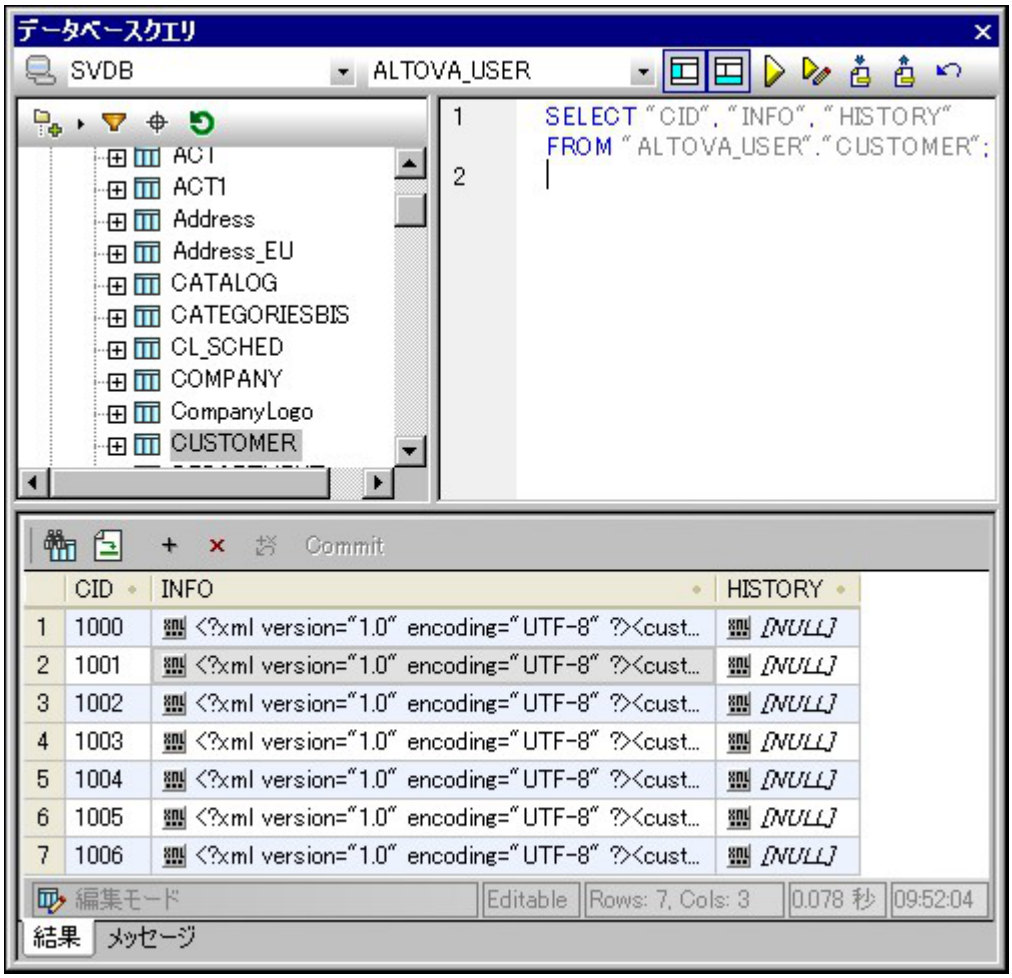
[フィルター] アイコンにより、特定の種類のメッセージの表示の切り替えをオンまたはオフします。「次」ならびに「前」アイコンにより、選択箇所を移動することができます。表示されているメッセージは、子コンポーネントを含めて(または含めないで)クリップボードにコピーすることができ、その後ドキュメントへの貼付けを行うことができます。[検索] 機能により、検索用語を指定し、この用語をリスト内上下に検索することができます。[クリア] アイコンにより、レポートペインの内容をクリアすることができます。

24.10.2.2 XML スキーマの割り当て

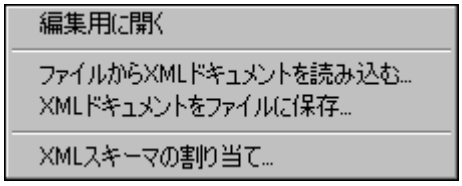
XML スキーマの割り当てコマンドにより、データベースクエリウィンドウの結果グリッドから、開かれている XML ファイルへスキーマを割り当てることができます。割り当てが行われると、そのスキーマに対して XML ファイルの検証を行うことができます。DB に対する割り当ては、XML ファイルが XMLSpy にて保存されたときに行われます。

DB XML ファイルを編集のために開く

データベースクエリウィンドウにてクエリが作成され、XML DB に対して編集を行うために実行されると、データベースクエリウィンドウの下部にある結果グリッドからデータベース内部にある XML ファイルへアクセス、そして編集することができます（以下のスクリーンショットを参照）。



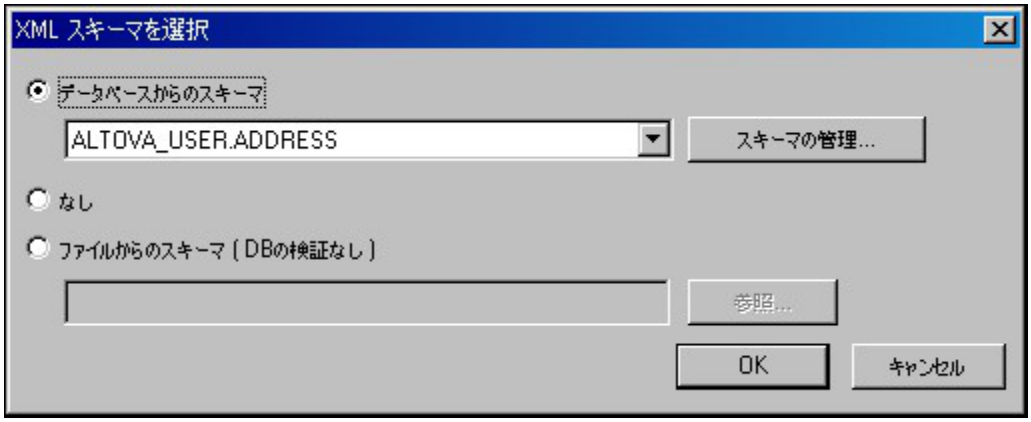
XML アイコン をクリックすることで、以下のメニューが表示されます。



「開いて編集」コマンドを選択すると、そのXMLドキュメントがXMLSpy で開かれ、編集を行うことができます。

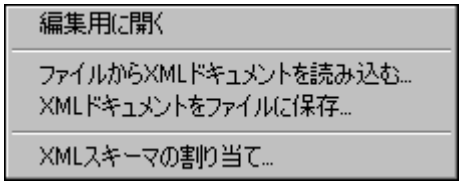
スキーマを DB XML ファイルへ割り当てる

DB XML ファイルが編集のためにXMLSpy で開かれると、「IBM DB2 | XML スキーマの割り当て」コマンドが有効になります。XMLドキュメントがXMLSpy でアクティブになっている状態で、「XML スキーマの割り当て」コマンドをクリックすると、XML スキーマを選択ダイアログが表示されます(以下のスクリーンショットを参照)。



(データベースからのスキーマコンボボックスにあるドロップダウンリストにて表示される)データベースに保管されている中からスキーマを選択することができる他にも、参照ボタンから外部ファイルを選択することもできます。「OK」をクリックすると、スキーマがXML ファイルへ割り当てられます。割り当て情報はXML ファイルへは書き込まれないことご注意してください。ファイルを開く際にデータベースクエリツールバーにあるXML の変更を自動コミットアイコン が有効になっていた場合、XMLSpy にてXML ファイルが保存されるとともに、スキーマの割り当てがデータベースへ保存されます。スキーマの割り当て情報はデータベースへ書きこまれ、XML ファイルへは書き込まれないことご注意してください。

メモ: データベースクエリウィンドウの結果グリッドには、XML スキーマの割り当てコマンドが用意されており、スキーマを DB XML ファイルへ割り当てることもできます(以下のスクリーンショットを参照)。



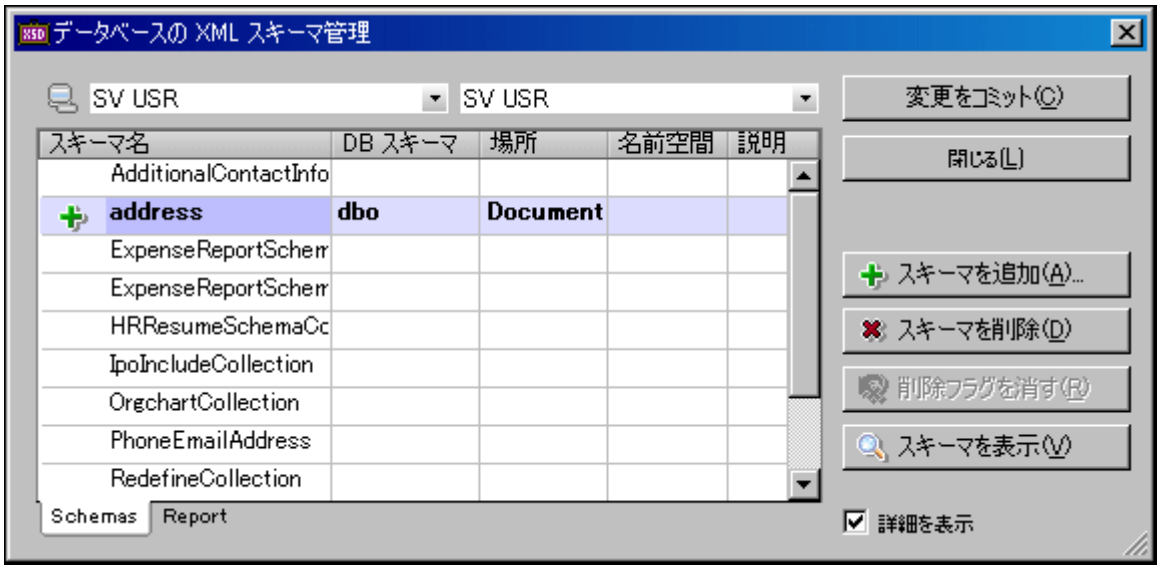
これら2つの XML スキーマ割り当てコマンドにおける違いは、「DB | IBM DB2」メニューのコマンドでは、XML ファイルの編集をしながらも XML スキーマの割り当てを行うことが可能で、XML ドキュメントの編集中にスキーマの割り当てを変更し、そのまま XML ドキュメントの検証を行うことが可能だということです。

24.10.3 SQL Server

SQL Server メニューアイテムには、XML スキーマの管理コマンドが収められています。

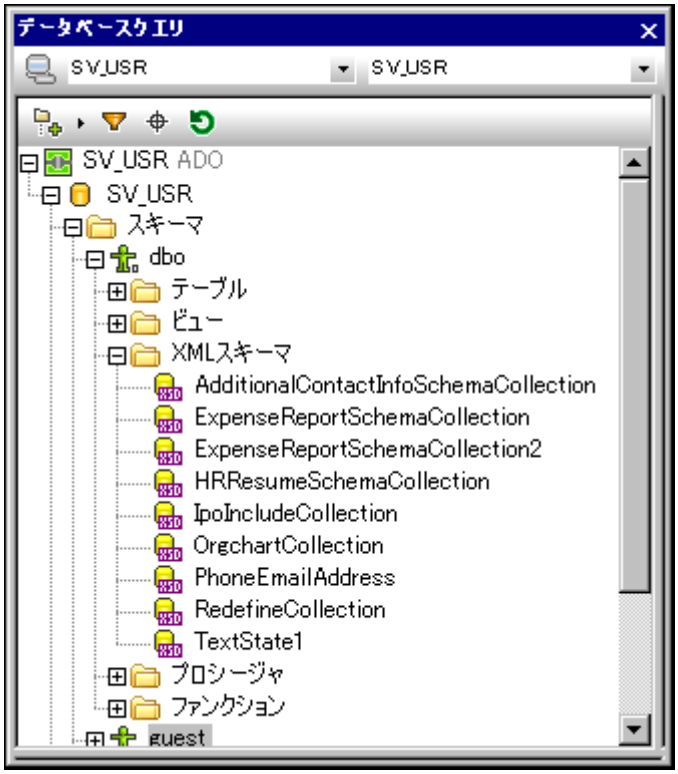
24.10.3.1 Manage XML Schemas

データベースの XML スキーマ管理ダイアログでは、XML データベースのスキーマリポジトリへ XML スキーマを追加したり削除することができます。データベースへの接続が行われると、XMLSpy に搭載されているデータベースの XML スキーマ管理ダイアログにて XML スキーマの管理を行うことができます。



ダイアログボックスに表示されているクイック接続アイコン を使用することで、クイック接続ウィザード 574 からデータソースへの接続することができます。1 つ以上の接続が現在存在する場合、必要な接続は左側のコンボボックスから選択することができます。必要なルートオブジェクトをコンボボックスの右側から選択することができます。ルートオブジェクトのためのレポジトリ内にある全ての XML スキーマは、ダイアログボックスに表示されています。名前、場所、各スキーマの名前空間のためのオプションがリストされます。

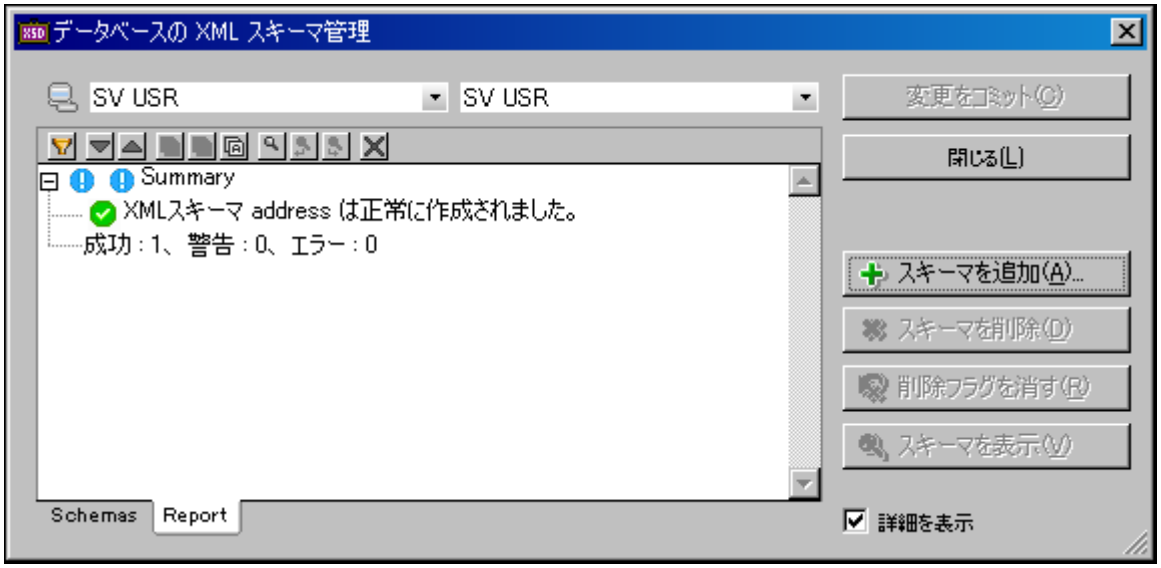
保管されているスキーマを、データベースクエリウィンドウ内で閲覧することができますが、管理することはできません（下のスクリーンショット）。スキーマを管理するには、データベースのための XML スキーマ管理ダイアログを使用します。



XML スキーマ管理ダイアログで以下を行うことができます:

- **スキーマの追加** ボタンを使用してスキーマを追加します。選択されたスキーマは、リストに追加され、追加のためにマークされます。
- **「スキーマをドロップ」** ボタンを使用して、リスト内のスキーマを削除のためにマークすることができます。**「削除フラグを消す」** ボタンを使用すると、ドロップフラグが削除されます。
- **スキーマをビュー** ボタンをクリックして、スキーマビュー内で選択されたスキーマを開きます。
- **「変更をコミット」** ボタンを使用して追加とドロップ(削除) の変更をコミットします。

変更がコミットされた後、レポートタブ内でコミットアクションに関するレポートを確認することができます(下のスクリーンショット)。



24.10.4 Oracle XML DB

XMLSpy はOracle XML Db データベースへの接続およびクエリを許可しています。

次のデータベース機能がサポートされています:

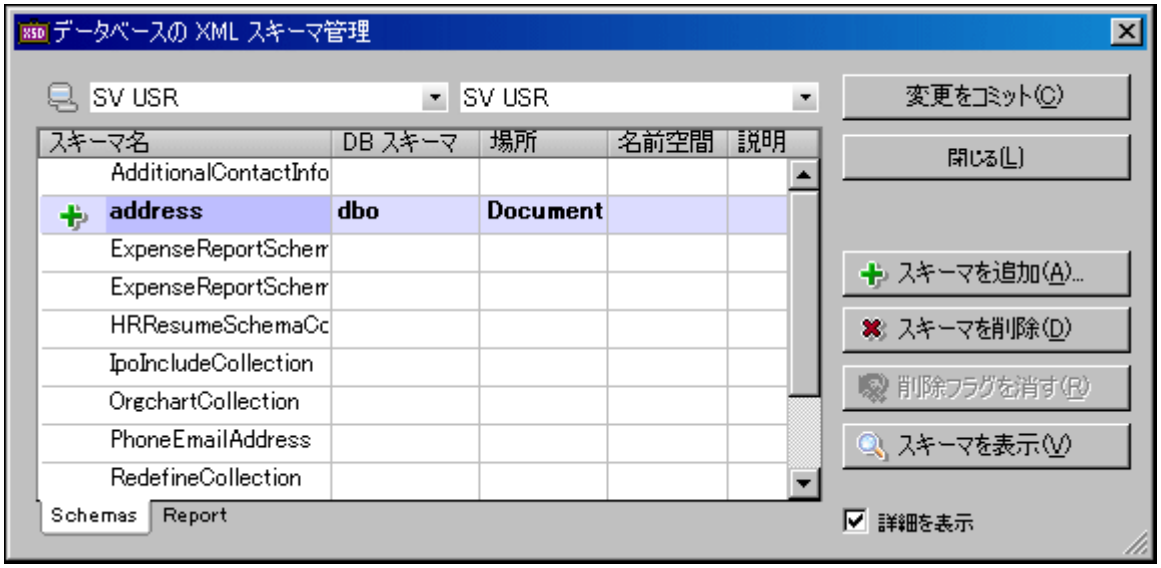
- XML スキーマを Oracle XML Db に追加および登録します。Oracle XML DB クライアントを XMLSpy を使用してスキーマを登録するためにインストールする必要があります。
- スキーマを開き削除します
- XPath ステートメント (DBUri) を使用したQuery データベース
- (W ebDAV を使用した)参照 XML ドキュメント
- データベース内に保存されたスキーマをベースにして作成された XML ドキュメント

一般的なインストールのプロセス:

- XMLSpy をダウンロードし、インストールします。
- Oracle サーバーを(必要な場合)インストールします。
- Oracle データベースを作成します

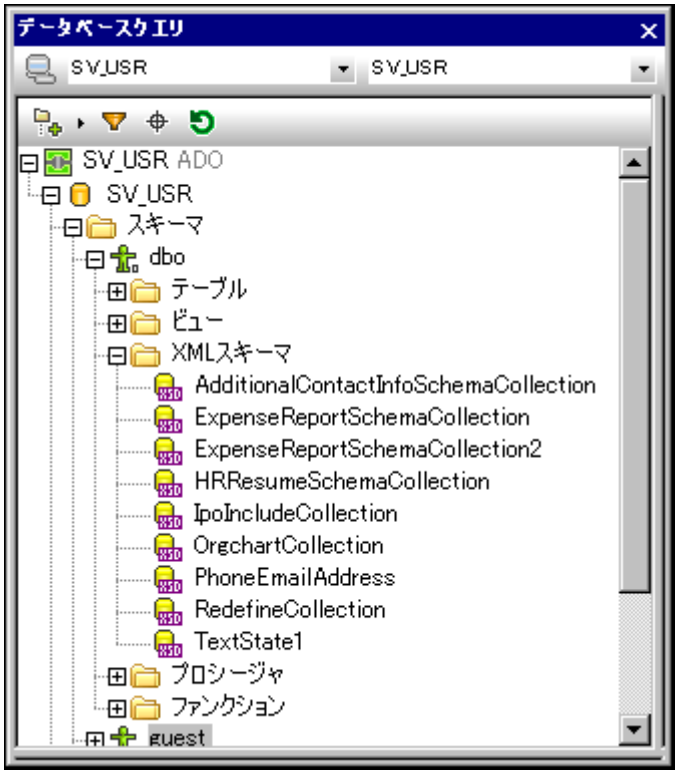
24.10.4.1 XML スキーマの管理

データベースの XML スキーマ管理ダイアログでは、XML データベースのスキーマリポジトリへ XML スキーマを追加したり削除することができます。データベースへの接続が行われると、XMLSpy に搭載されているデータベースの XML スキーマ管理ダイアログにて XML スキーマの管理を行うことができます。



ダイアログボックスに表示されているクイック接続アイコン を使用することで、クイック接続ウィザード 574 からデータソースへの接続することができます。1つ以上の接続が現在存在する場合、必要な接続は左側のコンボボックスから選択することができます。必要なルートオブジェクトをコンボボックスの右側から選択することができます。ルートオブジェクトのためのレポジトリ内にある全てのXML スキーマは、ダイアログボックスに表示されています。名前、場所、各スキーマの名前空間のためのオプションがリストされます。

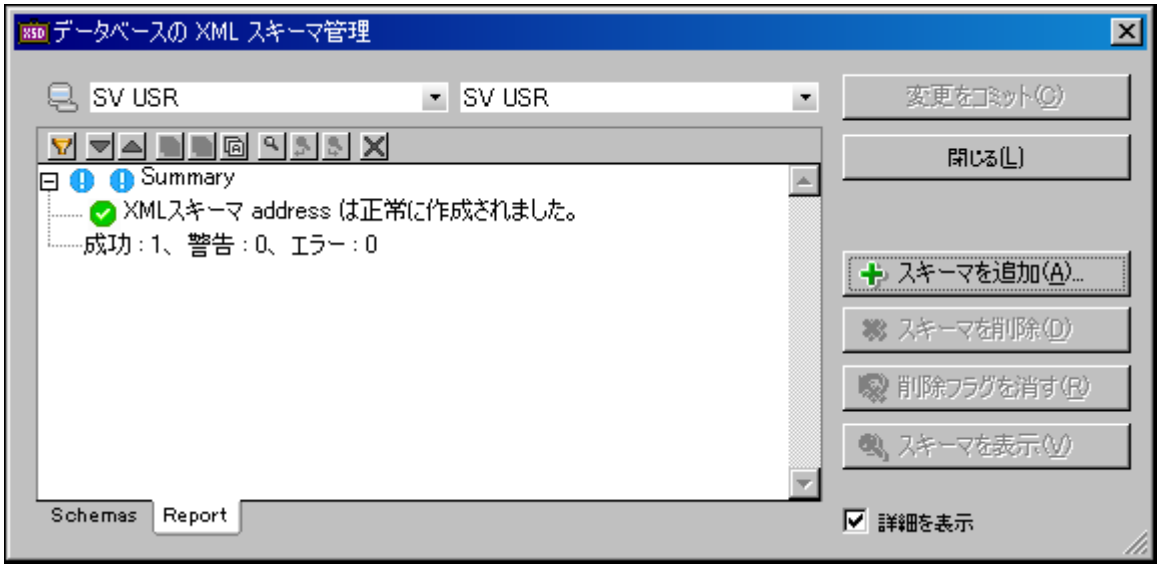
保管されているスキーマを、データベースクエリウィンドウ内で閲覧することができますが、管理することはできません（下のスクリーンショット）。スキーマを管理するには、データベースのためのXML スキーマ管理ダイアログを使用します。



XML スキーマ管理ダイアログで以下を行うことができます:

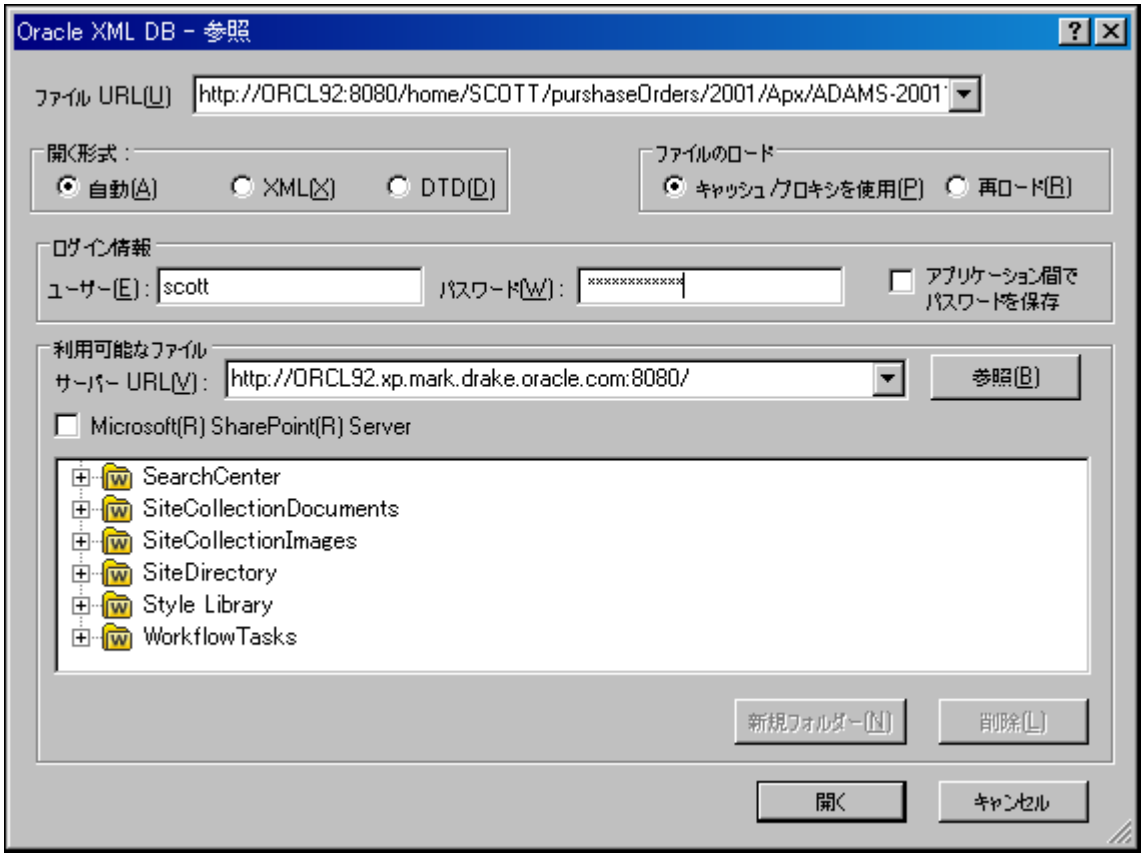
- **スキーマの追加** ボタンを使用してスキーマを追加します。選択されたスキーマは、リストに追加され、追加のためにマークされます。
- **「スキーマをドロップ」** ボタンを使用して、リスト内のスキーマを削除のためにマークすることができます。**「削除フラグを消す」** ボタンを使用すると、ドロップフラグが削除されます。
- **スキーマをビュー** ボタンをクリックして、スキーマビュー内で選択されたスキーマを開きます。
- **「変更をコミット」** ボタンを使用して追加とドロップ(削除) の変更をコミットします。

変更がコミットされた後、レポートタブ内でコミットアクションに関するレポートを確認することができます(下のスクリーンショット)。



24.10.4.2 参照 Oracle XML ドキュメント

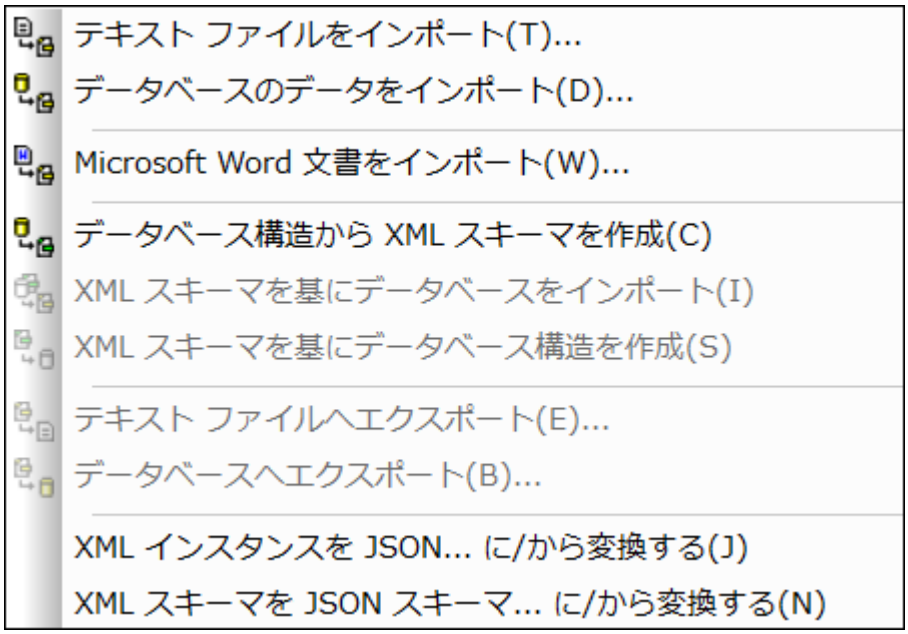
このコマンドにより、サーバー上で使用することのできる XML ドキュメントを参照することができます。以前にクエリされたデータベースまたはリストされたスキーマが存在する場合、サーバーの詳細は、自動的に入力されます。それ以外の場合は、手動で情報を入力します。



特定のXML ファイルをビューするために、ツリービューを使用します。ツリービューで開くファイルをダブルクリックします。ファイル を「開く」をクリックしても同じ結果が得られます。「**新規のフォルダー**」ボタンは、新規のフォルダーを追加します。「**削除**」ボタンは、現在選択されているXML ファイルを削除します。

24.11 変換メニュー

変換 メニューでは、以下に示される操作を行うためのパワフルなデータ交換機能が提供されます:

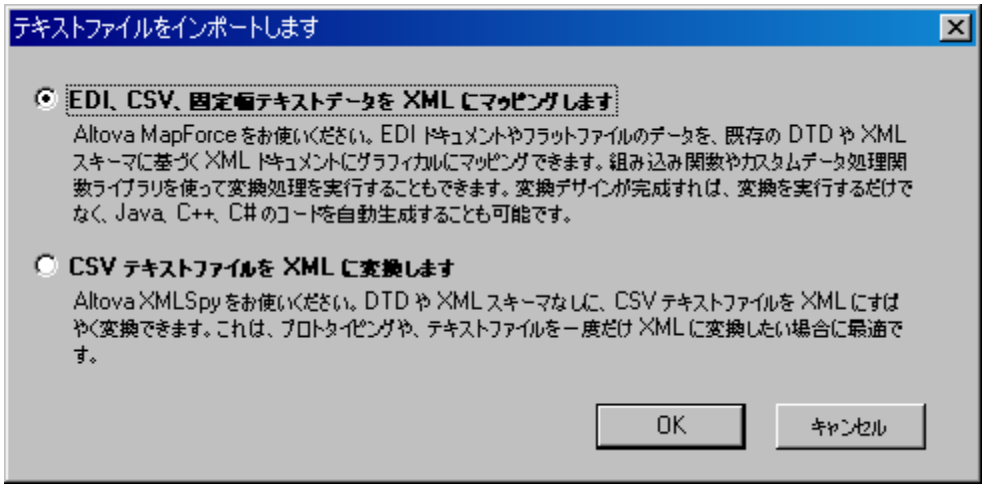


- テキスト、ワードプロセッサードキュメント、データベース、XML ファイルのインポートならびにエクスポート
- 既存の XML スキーマをベースにした[データベースのインポート](950)
- 既存データベースの構造をベースに [XML スキーマの作成](946)
- 既存の XML スキーマをベースに [データベース構造を作成](951)。
- [XML インスタンスと JSO インスタンス間](959) と[XML スキーマと JSON スキーマ間の](962) の変換。

24.11.1 テキストファイルのインポート

このコマンドにより、**構造化テキスト** ファイルを XMLSpy へ**インポート**し、直ちに XML フォーマットへ変換することができます。古いシステムからレガシーなデータをインポートするためにこの機能を利用することができます。以下にテキストファイル内のデータを XML ドキュメントとしてインポートする為の手順を記します。

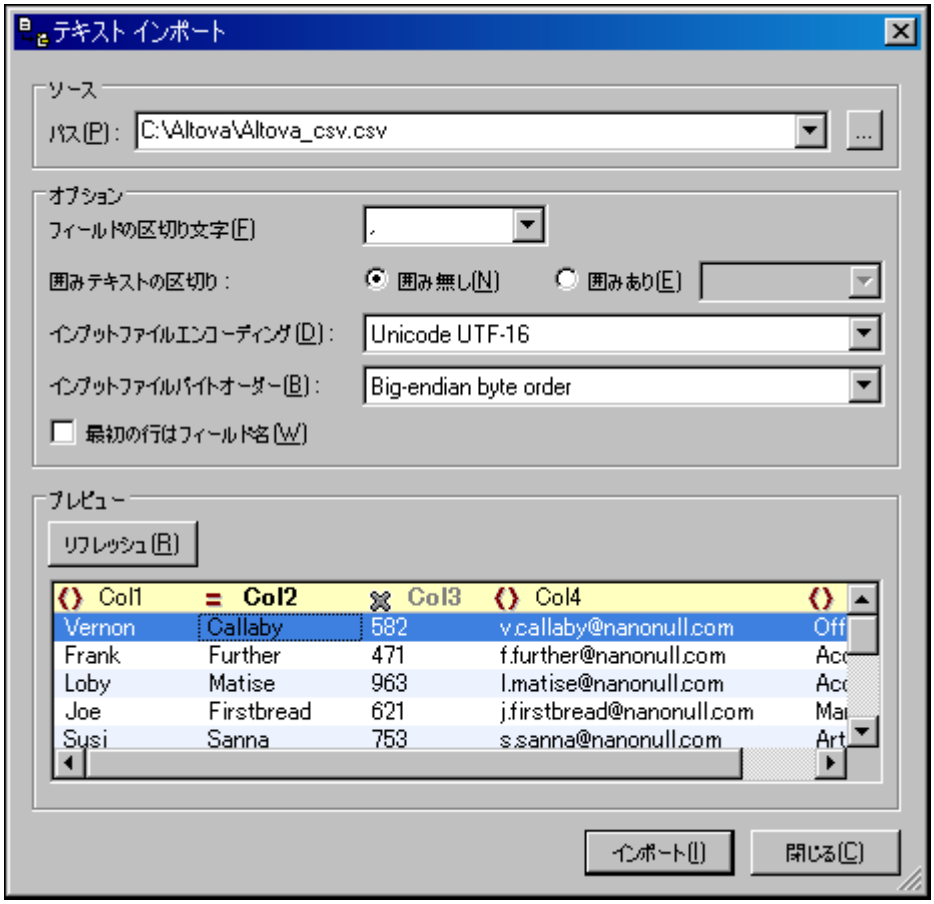
1. 「**変換 | テキストファイルのインポート**」を選択します。以下のダイアログが表示されます:



2. 以下から選択を行います:

- EDI、CSV、固定長テキストデータを XML へマッピング(このオプションを選択するには Altova MapForce がインストールされていなければなりません)
- CSV テキストファイルを XML へ変換

3. 「OK」をクリックします。テキストインポートダイアログが表示されます(以下のスクリーンショットを参照)。以下にこのダイアログの使用方法を記します。



パス

インポートするファイルへのパスをパス テキストボックスへ入力するか、テキストボックスの右側にある参照ボタンを使用してファイルを選択します。ファイルが選択されると、XML ファイルのプレビューが、グリッドビュー形式で*プレビュー* ペインに表示されます。このダイアログにおけるオプションの変更は、プレビューへ直ちに反映されます。

区切り文字

テキストファイルをインポートするには、ファイル内部のカラムやフィールドを分割している区切り文字を指定する必要があります。標準的な行の区切り（CR、LF、または CR+LF）は <%XMLSPY% により自動的に識別されます。

文字列の引用符

レガシーなシステムからエクスポートされたテキストファイルの中には、数値データとの違いを明確にするために、文字列の値を引用句により囲んでいるものもあります。このような場合、ファイルの中で使用されている引用句を指定することで、データがインポートされた際にそれら引用句を自動的に削除することができます。

(C) 2015-2021 Altova GmbH

エンコード

データは（XML ドキュメントで通常使用される）ユニコード[1591] へ変換されるため、ファイル内で現在使用されているエンコーディングを指定する必要があります。

バイトオーダー

16ビットまたは32ビットのユニコード（UCS-2、UTF-16、または UCS-6）ファイルをインポートする場合、リトルエンディアンとビッグエンディアンバイトオーダーから選択を行うことができます。

最初の行はフィールド名

テキストファイル内にある最初の行にてフィールド名が記述されることもあります。この場合、このチェックボックスにチェックを入れてください。

プレビュー

プレビューペインでは、カラムヘッダーの名前をダブルクリックして編集を行うことで、カラム名を変更することができます。カラムヘッダーは、XML ドキュメントにおける要素または属性名となります。各カラムのヘッダーにあるアイコンをクリックすることで、カラムが要素なのか属性なのか、また XML ドキュメントにインポートするのかを選択することもできます。上のスクリーンショットでは、Col1 が要素、Col2 が属性としてインポートし、Col3 をインポートしないようにセットされています。

オプションの指定が終わったら、「**インポート**」をクリックします。インポートされたデータが XML ドキュメントに変換され、グリッドビューにて表示されます。

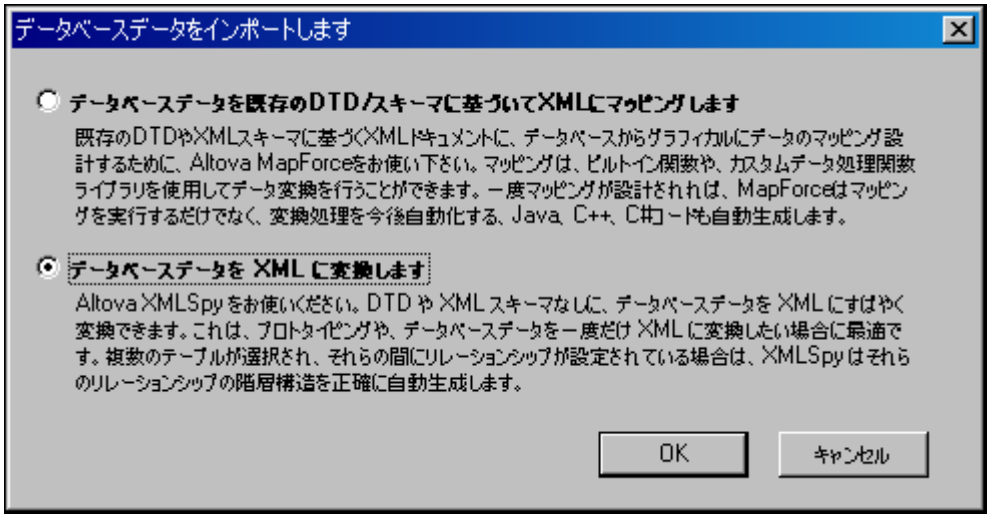
24.11.2 データベースのデータをインポート

「**データベースのデータをインポート**」コマンドにより、様々な種類のデータベースからデータを XML ファイルにインポートすることができます。インポートメカニズムは、以下のような2つのステップにより構成されます:

1. データベースへの接続が作成されます。詳細に関してはデータベースへ接続[573] を参照してください。
2. インポートするデータを選択[942] します。

以下の操作によりデータベースのデータをインポートします:

1. データベースのデータをインポートコマンドをクリックすると、データベースデータをインポートするためのダイアログが表示されます（以下のスクリーンショットを参照）。

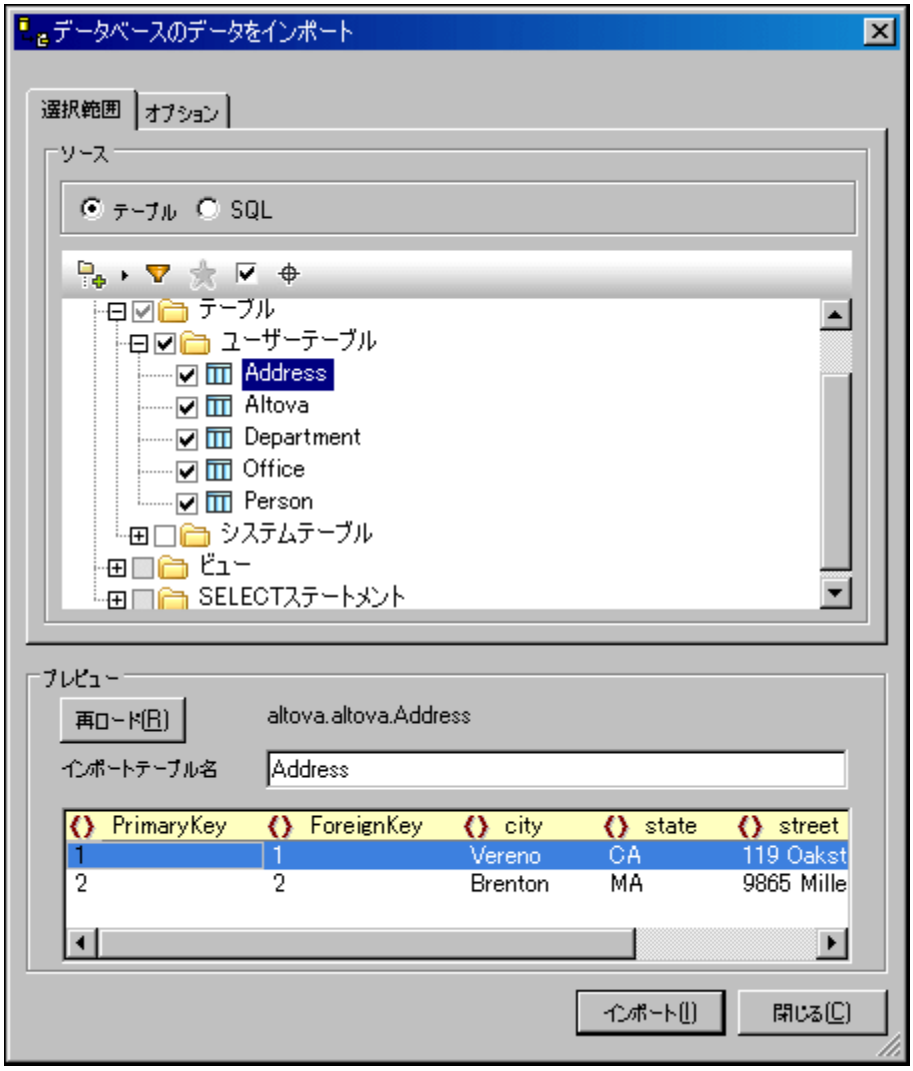


2. データベースのデータを XML に変換を選択して、OK をクリックします。データソースへの接続ダイアログが表示されます。データベースを既存の DTD/スキーマに基づいて XML にマッピングオプションを使ってマッピングを行うには、Altova MapForce が必要になります。
3. データソースへの接続ダイアログにて、データベースへの接続を確立します。異なる種類のデータベースに対する接続の方法については、データベースへ接続 573 セクションを参照ください。
4. データベースへの接続が確立されると、データベースのデータをインポートダイアログにて表示されているタブやウィンドウにより、データベースにあるデータのインポートを始めます。以下にこれらオプションに関する説明を記述します。オプションの設定を完了すると、インポートボタンをクリックしてデータベースのデータをインポートします。

データの選択とインポートオプション

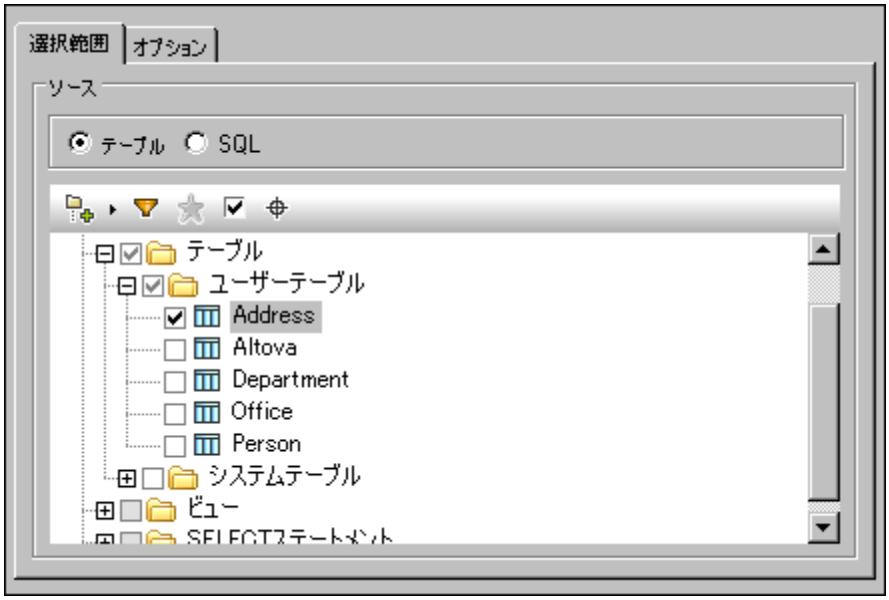
インポートオプションを指定するデータベースのデータをインポートダイアログは、2つのパートにより構成されています:

- 選択範囲とオプションという2つのタブがある上部パート。
- データの選択範囲とインポートオプションに従ったデータを表示するプレビューウィンドウがある下部パート。



選択範囲タブ

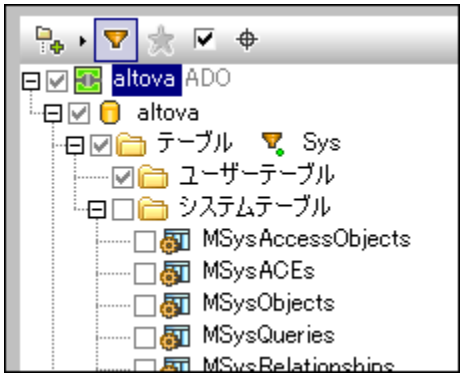
(上のスクリーンショットで示される)選択範囲タブでは、(以下のスクリーンショットにある)ソースペインによりデータベース上のテーブルや、目的のテーブルを選択するためのSQL ステートメントが表示されます。



テーブルビューでは、テーブルのチェックボックスにチェックを入れることで、インポートするテーブルをデータベースから選択することができます（上のスクリーンショットを参照）。テーブルのコンテンツはプレビューペインに表示されます。テーブルの選択は、プレビューペインにて更にフィルタリングすることもできます。

データベース構造の表示方法を変更したり、フィルタリングを行うこともできます。ソースペインにあるレイアウトアイコン により、データベースオブジェクトを、(i) オブジェクトの型をベースにしたフォルダー、(ii) スキーマフォルダー無しで、オブジェクトの型をベースにしたフォルダー、(iii) フォルダー無しの階層構造、(iv) 他のテーブルとのリレーションシップをベースにしたテーブルのカテゴリーにより管理することができます。

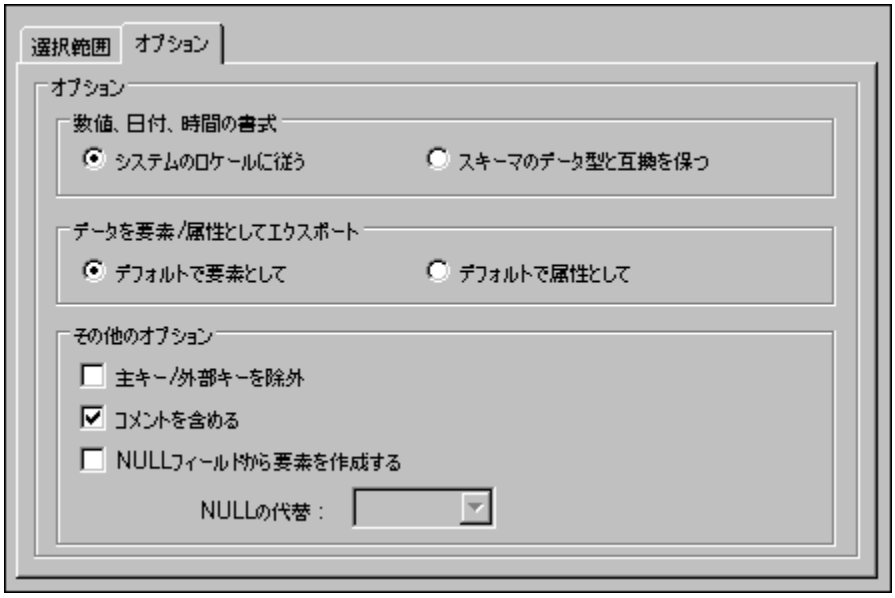
フォルダーコンテンツのフィルター アイコンにより、選択されたフォルダーに対してフィルタリングを行うことができます（以下のスクリーンショットにあるテーブルフォルダーを参照）。テーブルの隣にあるフィルターアイコンをクリックすることで、フィルタリングの種類を選択することができます。以下のスクリーンショットでは、Sys というテキストが名前に含まれているオブジェクトだけを表示するようにフィルタリングが行われています。



オブジェクトロケーターアイコン をクリックすると、検索フィールドの様に動作するテキストフィールドが表示されます。テキスト文字列を入力すると、入力された文字列を含む全てのオブジェクトが表示されます。ドロップダウンリストにて表示されているオブジェクトを選択することで、ツリー内の対応するオブジェクトがハイライトされます。

オプションタグ

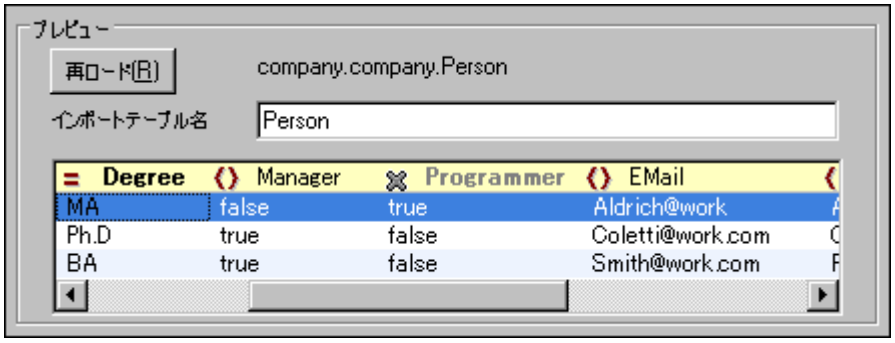
（以下のスクリーンショットで示される）オプションタブでは、数値、日付、時間の値をインポートする方法を指定することができます。要素や属性としてデータをインポートするのか、コメントやNULL フィールドもインポートの中に含まれるかを指定することができます。



NULL フィールドをインポートする場合、その代替となるXML の値を入力することができます。

プレビューペイン

（以下のスクリーンショットで示される）プレビューペインでは、選択範囲タブにて現在選択されているテーブルの構成が表示されます。選択範囲タブにて新しいテーブルが選択された場合、**再ロード**ボタンによりプレビューが更新されます。



要素または属性としてフィールドをインポートするように指定することができ、また、列名の左横のシンボルをクリックして、インポートされないように指定することもできます。要素、属性、またはオプションの無視から選択することができます。上のスクリーンショットでは、city フィールド、が例えば、要素としてインポートされるようにセットされていますが、street フィールドは属性としてインポートされるように設定されています。

データ型の変換

データベースのデータ型からXML スキーマのデータ型への変換に関する情報については、付録 1578 を参照ください。

24.11.3 Microsoft Word ドキュメントをインポート

このコマンドにより任意のWord ドキュメントをインポートし、XML フォーマットへ変換することができます。このオプションを使用するには、お使いのコンピューターにMicrosoft Word またはMicrosoft Office がインストールされていなければなりません。

このコマンドを選択すると、ファイルを開くダイアログが表示されます。インポートするWord ドキュメントを選択してください。
XMLSpy により、CSS スタイルシートとともにXML ドキュメントが生成されます。Word における各段落から、Microsoft Word 内の段落スタイルに対応した名前のXML 要素が生成されます。

24.11.4 データベース構造から XML スキーマを作成

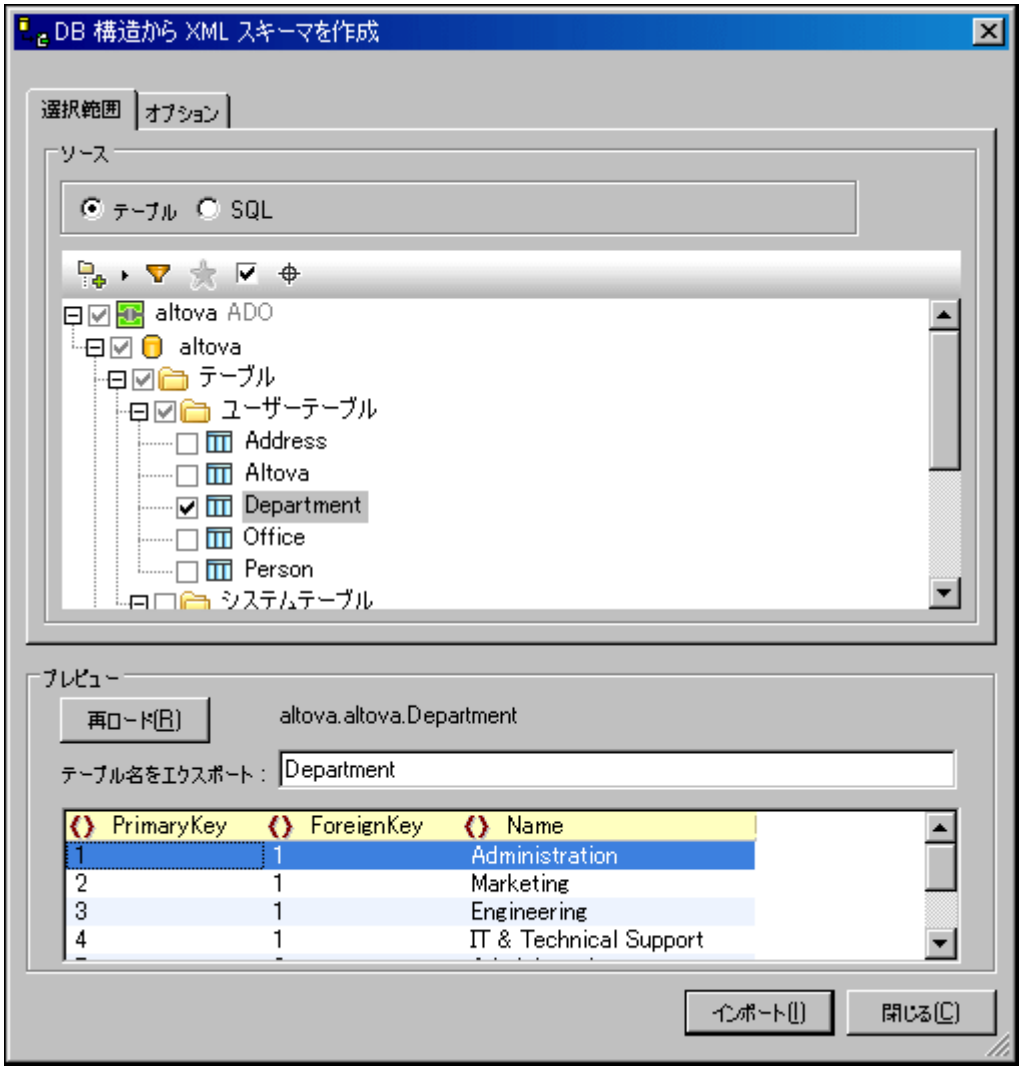
「**データベース構造から XML スキーマを作成**」コマンドにより、様々な種類のデータベース構造からXML スキーマを作成することができます。XML スキーマの作成メカニズムは、2つのステップにより構成されます:

1. データベースへの接続が確立されます。
2. データベースのデータ選択や XML スキーマのオプションを指定します。

以下の操作により、データベース構造からXML スキーマを作成することができます:

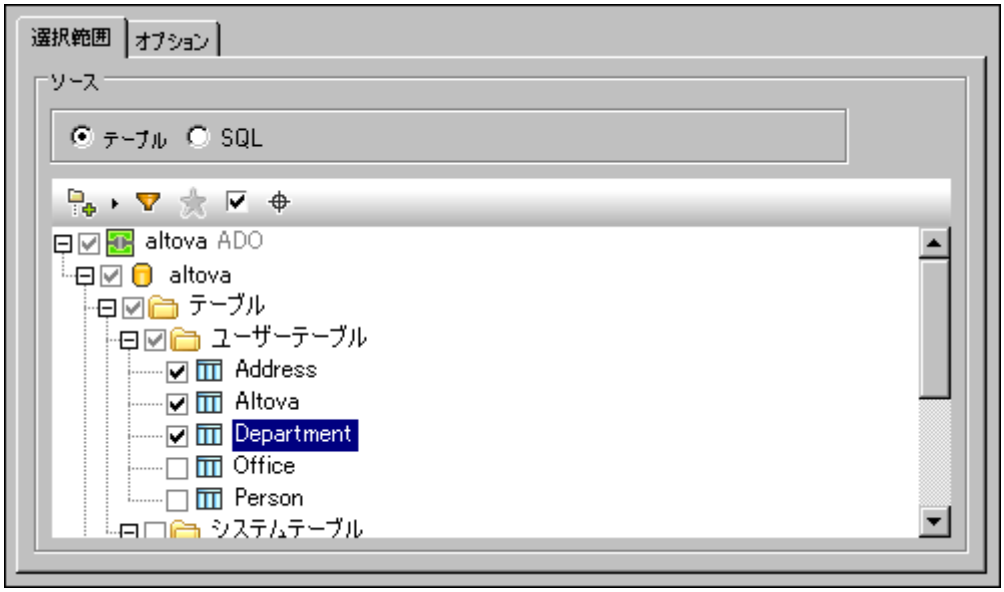
1. **データベース構造から XML スキーマを作成**コマンドをクリックすると、データベース接続ウィザードボックスが表示されます。
2. データベースへの接続ウィザードにてデータベースへの接続を確立します。異なる種類のデータベースに対する接続の方法についてはデータベースへ接続 573 セクションを参照ください。
3. データベースへの接続が確立されると、DB 構造から XML スキーマを作成ダイアログが表示され、タブやペインによりインポートするデータベース構造を選択することができます（以下のスクリーンショットを参照）。オプションについては以下を参照ください。**インポート**ボタンをクリックすることで、データベースのデータがインポートされます。

(C) 2015-2021 Altova GmbH



選択範囲タブ

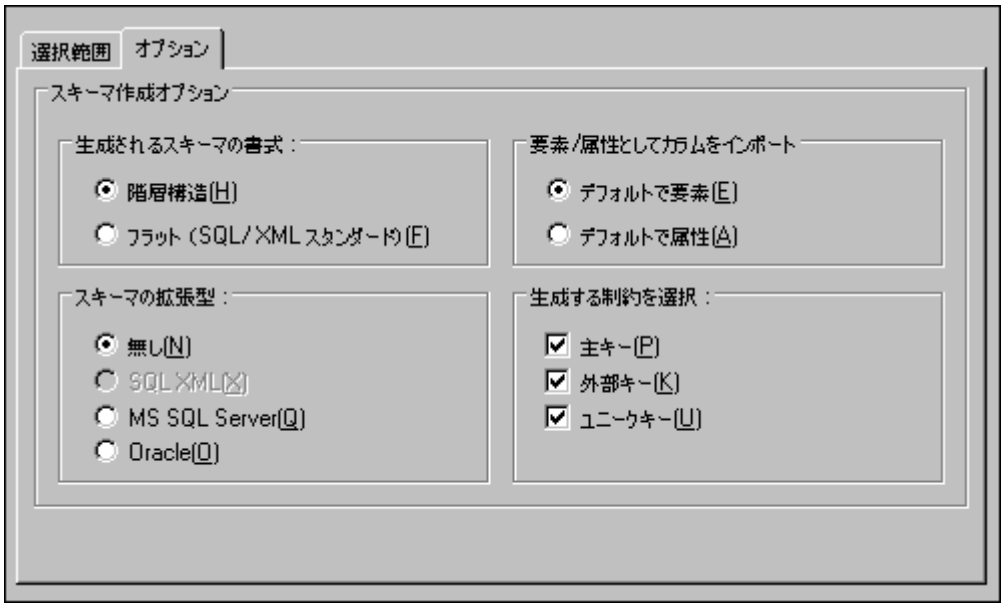
（上のスクリーンショットで示される）選択範囲タブでは、（以下のスクリーンショットにある）ソースペインによりデータベース上のテーブルや、目的のテーブルを選択するためのSQL ステートメントが表示されます。.



テーブルビューでは、テーブルのチェックボックスにチェックを入れることで、インポートするテーブルをデータベースから選択することができます（上のスクリーンショットを参照）。テーブルのエンテンツはプレビューペインに表示されます。テーブルの選択は、プレビューペインにて更にフィルタリングすることもできます。データベースデータのインポート 941 のセクションに記述されている方法を使用することで、ソースペインの表示方法を変更することもできます。

オプションタグ

（以下のスクリーンショットにある）オプションタブでは、スキーマの書式、その拡張型、カラムを要素または属性としてインポートするのか、スキーマにてデータベースの制約を生成するかを指定することができます。



スキーマの書式：フラット（SQL/XML 標準）か階層構造スキーマを選択することができます。

- **フラット**スキーマモデルはXML-Related Specification (SQL/XML) という名前のISO-ANSI 作業草案(これ以降は「SQL/XML 作業草案」と表記)をベースにしています。SQL/XML 作業草案では、データベースを XML へマッピングする方法が定義されます。スキーマにおいてリレーションシップは同一性制約を使って定義され、要素への参照は行われません。そのため、スキーマはデータベースのツリー型ビューにも似たフラットな構造となります。ANSI ストアで仕様を購入することができますSQL/XML 作業草案に関するより詳細な情報は、以下のリンクを参照ください: www.sqlx.org。
- **階層構造**スキーマモデルでは、何らかのオブジェクトに依存しているテーブルがインデントされた子要素としてコンテンツモデルに表示され、ツリービューの様な形式でテーブルの依存関係が視覚的に表示されます。テーブルの依存関係は同一性制約タブにも表示されます。スキーマにおいてテーブルはグローバル要素として表示され、テーブルのカラムはこれらグローバル要素の子要素または属性となります(カラムを要素とするか属性とするかはユーザーにより決定されます)。リレーションシップは階層構造形式で作成され、あるテーブルにある外部キーのフィールドは、そのテーブルを表すグローバル要素への参照となります。

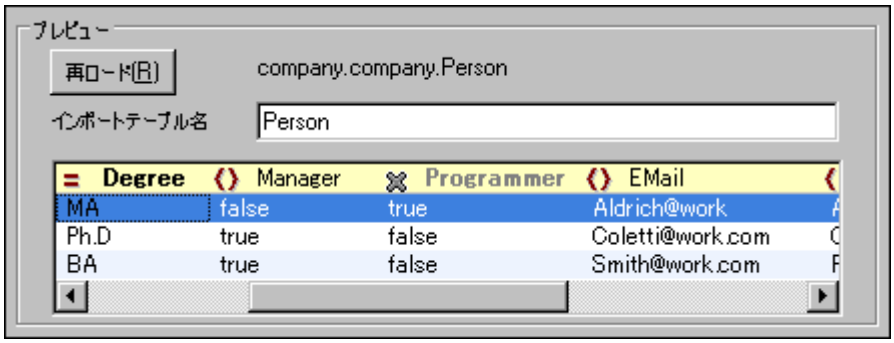
スキーマの拡張型: スキーマ拡張情報とはデータベースから読み取られ、スキーマ内部では注釈データまたは属性として埋め込まれる情報のことです。スキーマを生成するにあたり、4つの拡張型オプションから選択を行うことができます: (i) 拡張情報無し、(ii) SQL/XML 拡張、(iii) MS SQL Server 拡張、(iv) Oracle 拡張。それぞれのオプションについて以下に記述します:

- 無し: データベースから新たに取得する情報はありません。
- SQL XML: :SQL/XML 拡張は、フラット構造でスキーマの生成を行うときのみ使用することができます。拡張情報は SQL/XML 作業草案に従ったかたちで注釈内に保管されます。(INCITS/ISO/IEC 9075-14-2008).
- **MS SQL Server:** Microsoft SQL Server を選択することで、SQL Server 拡張が生成されます。詳細については、SQL サーバーブックオンラインとMSDN のXSD スキーマの注釈付けについてを参照ください。以下にある注釈データのサブセットがスキーマにて生成されます: sql:relation、sql:field、sql:datatype、sql:mapped。
- **Oracle:** :Oracle データベースを使用していると、Oracle 拡張がデフォルトで選択されます。拡張情報は属性内に保管されます。これら属性に関する情報については、Oracle のオンラインドキュメント. を参照ください。以下にある属性のサブセットが現在生成されます: SQLName、SQLType、SQLSchema。.

メモ: SQL Server ならびに Oracle 拡張は、それぞれ対応するデータベース以外のデータベースに対して生成することができます。この機能は、別のデータベースにて作業しながら、将来 SQL Server または Oracle にて処理されるスキーマを生成するような場面で利用することができます。

プレビューペイン

(以下のスクリーンショットで示される)プレビューペインでは、選択範囲タブにて現在選択されているテーブルの構成が表示されます。選択範囲タブにて新しいテーブルが選択された場合、**再ロード**ボタンによりプレビューが更新されます。



要素または属性としてフィールドをインポートするように指定することができ、また、列名の左横のシンボルをクリックして、インポートされないように指定することもできます。要素、属性、またはオプションの無視から選択することができます。上のスクリーンショットでは、`city` フィールド、が例えば、要素としてインポートされるようにセットされていますが、`street` フィールドは属性としてインポートされるように設定されています。

データ型の変換

データベースのデータ型からXML スキーマのデータ型への変換に関する情報については、付録 1578 を参照ください。

24.11.5 XML スキーマを元にデータベースをインポート

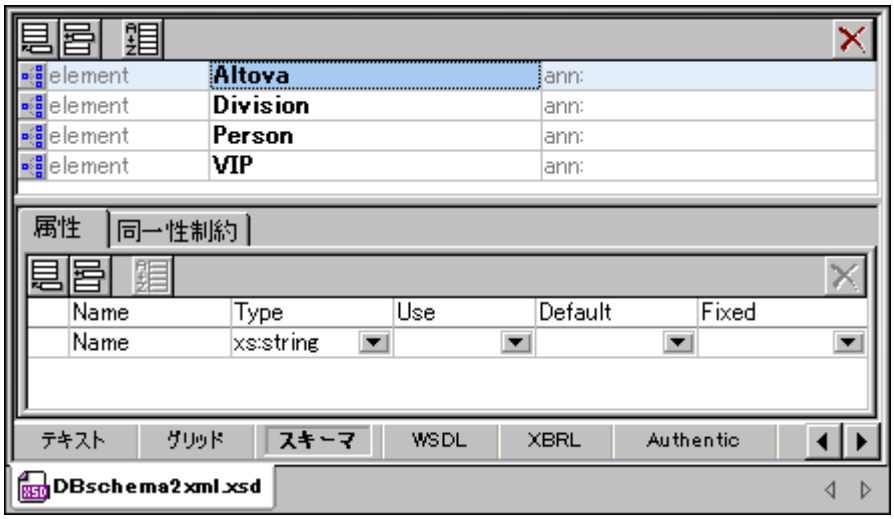
「XML スキーマを元にデータベースをインポート」コマンドにより、与えられた XML スキーマに従う形で妥当かつ、データベースからインポートされたデータを含む XML ドキュメントが作成されます。この機能は以下のデータベースにてサポートされます:

- Microsoft Access 2000 と2003
- Microsoft SQL Server
- Oracle
- MySQL
- Sybase
- IBM DB2

インポートされるデータはデータベースにて選択されているテーブルにより決定されます。スキーマビューにてアクティブになっているドキュメントが、インポートにて使用される XML スキーマとなり、データベースへの接続が行われます。インポートを行うテーブルを選択し、インポートをクリックします。データが XML ドキュメントへインポートされ、データのインポート時にアクティブだった XML スキーマに従う形でドキュメントは構成されます。

以下の例では、MS Access データベース上にあるデータが、スキーマビューでアクティブな XML スキーマに従ってにインポートされます。インポートを行うためのステップを以下に示します:

1. スキーマファイルをスキーマビューにて選択します（以下のスクリーンショットを参照）。



2. メニューコマンド「**XML スキーマを元にデータベースをインポート**」を選択します。データソースへ接続 574 ダイアログが表示されます。
3. Microsoft Access (ADO) オプションを選択し、**次へ**をクリックします。
4. **参照**をクリックし、データベースファイルを選択します。**次へ**をクリックします。

5. XML スキーマに基づく DB インポートダイアログにて、テーブルタグへ移動し、インポートを行うテーブル（例えば Altova）を選択し、「**インポート**」をクリックします。テーブルが XML ドキュメントへインポートされ、グリッドビューにて表示されます。

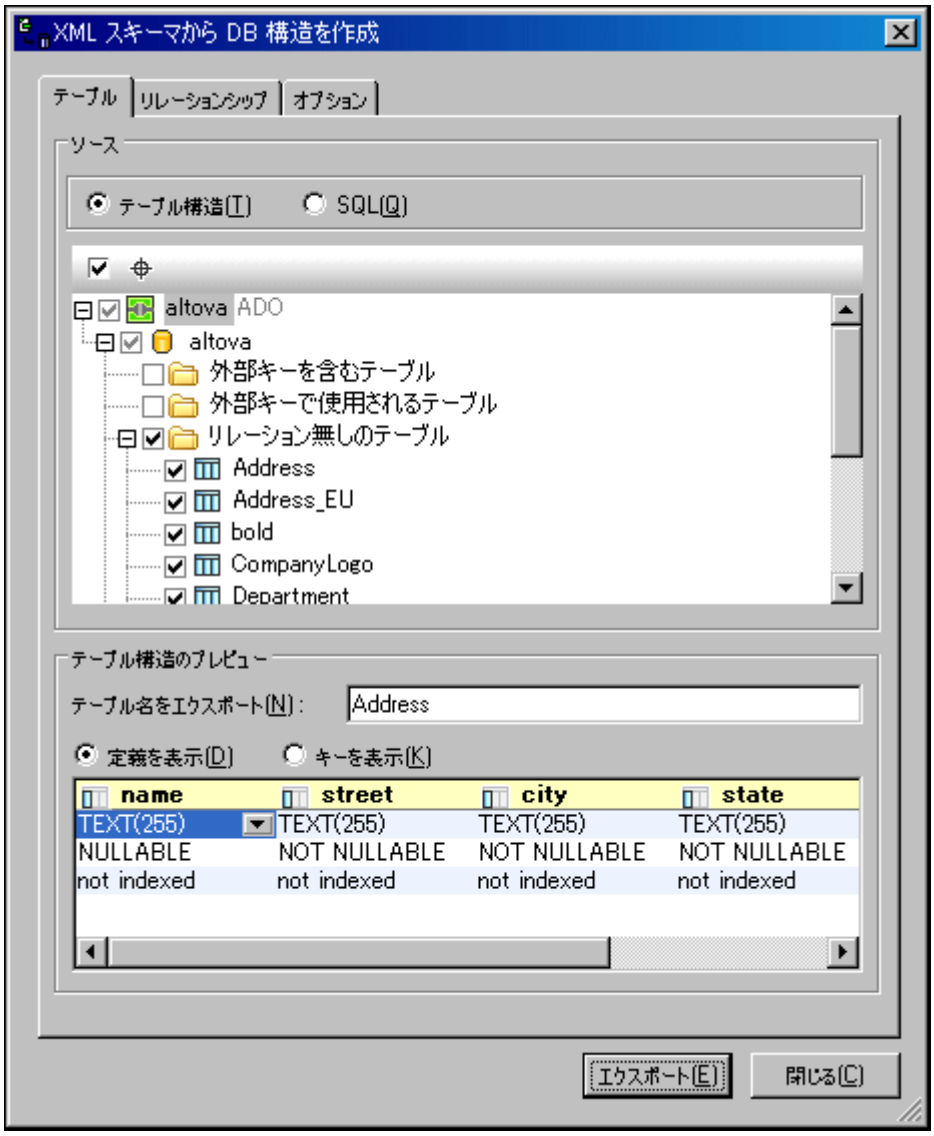
データ型の変換

データベースのデータ型からXML スキーマのデータ型への変換に関する情報については、付録[1578]を参照してください。

24.11.6 XML スキーマを元にデータベース構造を作成

XMLSpy では、既存のスキーマファイルを元に、空のデータベース(またはスケルトンデータベース)を作成することができます。以下に記される方法は、各種データベースに対して使用することができます。

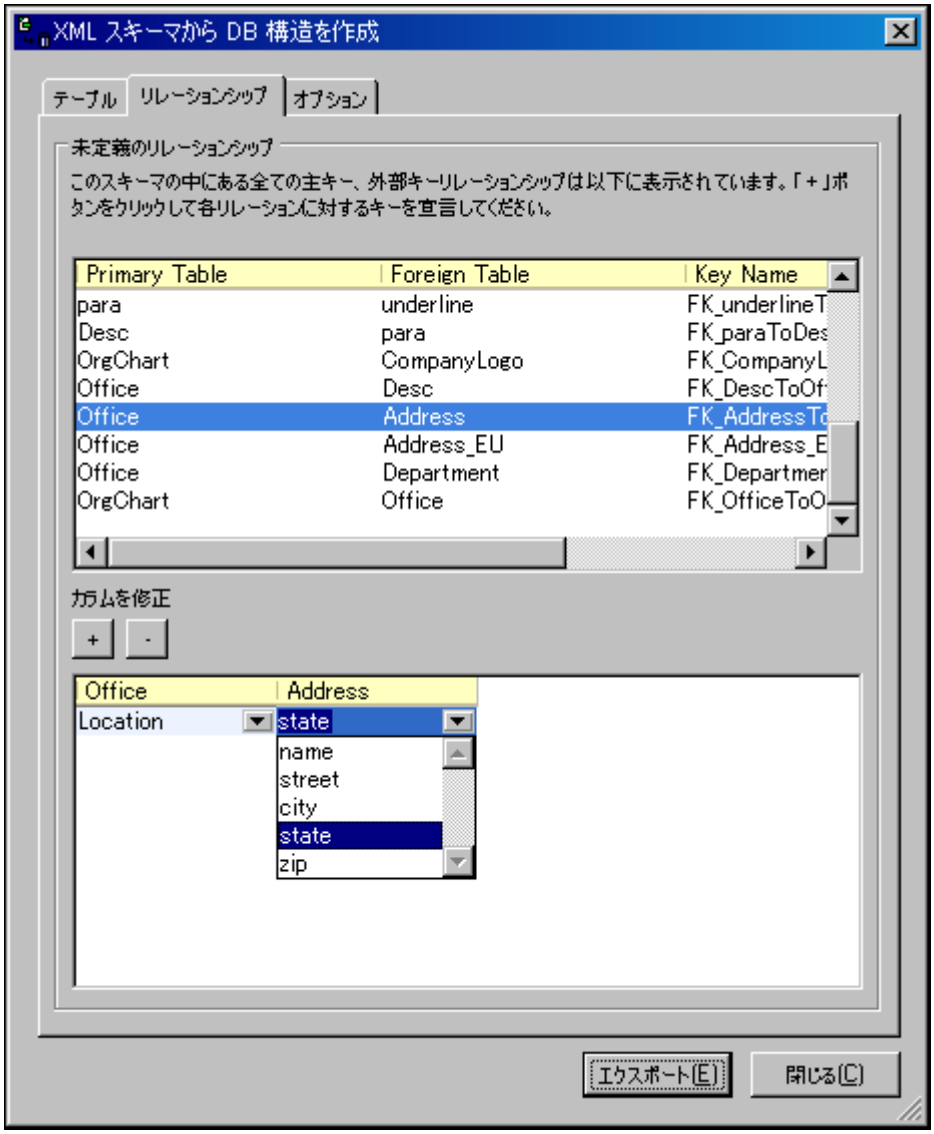
1. スキーマファイルをスキーマ/WSDL ビューにて開きます。
2. メニューコマンドから「**変換 | XML スキーマを元にデータベース構造を作成**」を選択します。データソースへの接続[574] ダイアログが表示され、データベース(DB) への接続を行います。
3. リレーショナルデータベースへの接続を行うには、データソースへの接続[573] のセクションに記述された方法を使用します。例えば、Microsoft Access データベースへ接続するには、Microsoft Access ラジオボタンを選択し、データベースを選択する操作を続けます。スキーマ構造の作成には、既存のデータ
4. XML スキーマから DB 構造を作成ダイアログにて、スキーマから作成されたテーブルがツリー構造で表示されます。例えば以下のスクリーンショットでは、Address テーブルが作成され、エクスポートされる対象として選択されています。エクスポートしないテーブルは、隣にあるチェックボックスやコンテキストメニューからチェックを外してください。



リレーションシップと共にDB テーブルを作成する

DB 構造の元となるXML スキーマにて同一性制約によるリレーションシップが定義されている場合、これらのリレーションシップが自動的に作成され、テーブル構造にて表示されます。外部キーを含むテーブルならびに外部キーで使用されるテーブル以下に、リレーションシップを含むテーブルが表示されます。

リレーションシップタブでは、テーブルリレーションシップを作成ならびに修正することができます。このタブでは、主キー/外部キーが取りうる全てのリレーションシップが表示されます(以下のスクリーンショットを参照)。



以下の操作により、リレーションシップを作成することができます:

1. 表示されている中から、1つの主キー/外部キーリレーションシップを選択してください。
2. ダイアログ下部にあるペインにてプラスボタンをクリックして、リレーションシップを作成します。
3. 2つのテーブルにあるドロップダウンリストから、目的のカラムを選択します。

リレーションシップを選択して、マイナスボタンをクリックすることで、リレーションシップを削除することができます。

データベース構造ならびに接続に関するメモ

同一性制約により定義されたスキーマ構造は、生成されるデータベースにも反映されます。XML スキーマから DB 構造を作成ダイアログにより作成されるデータベースの種類と制約、そして接続方法を以下のテーブルに記します。

	直接	ODBC を使用	ADO を使用

MS Access (2000 と2003)	OK *	OK	OK
MS SQL Server	OK *	OK	OK
Oracle	OK *	OK	OK
MySQL	-	OK *	OK ⌖
Sybase	-	OK *	OK
IBM DB2	-	OK *	OK

* *推奨される接続方法*
⌖ *MySQL: ODBC をベースにした ADO 接続を作成する場合、User またはシステム DSN の使用が推奨されます*
\- *未サポート*

XMLSpy では、階層構造ならびにフラット構造のスキーマ946 のマッピングが行われます。両者のフォーマットは、XMLSpy により自動的に認識されます。
フラット構造から SQL へのマッピングは2つの点で異なります:

- SQL Server DB、Oracle DB、または Sybase DB:
  上にあるデータベースに対してフラット構造で生成されたスキーマは、スキーマカタログ名が抽出され、生成された SQL スクリプトにて DB 名として使用されます。つまり、生成される SQL スクリプトを実行するには、ターゲット DB の名前をスキーマカタログ名と同じものにする必要があります。
- Access (2000 ならびに 2003)、MySQL、または DB2 DB:
  上にあるデータベースに対してフラット構造で生成されたスキーマは、SQL スクリプトが生成される時に、スキーマカタログ名を無視します。つまり、生成される SQL はログインしたターゲットデータベースに対して実行されます。

データ型の変換

データベースのデータ型から XML スキーマのデータ型への変換に関する情報については、付録1578を参照ください。

24.11.7 テキストファイルへエクスポート

「**変換 | テキストファイルへエクスポート**」コマンドにより、XML データをテキスト形式でエクスポートすることで、データベースやレガシーなシステムと、データの受け渡しを行うことができます。ダイアログは2つのパートから構成されます(それぞれ別々にスクリーンショットを以下に示します)。

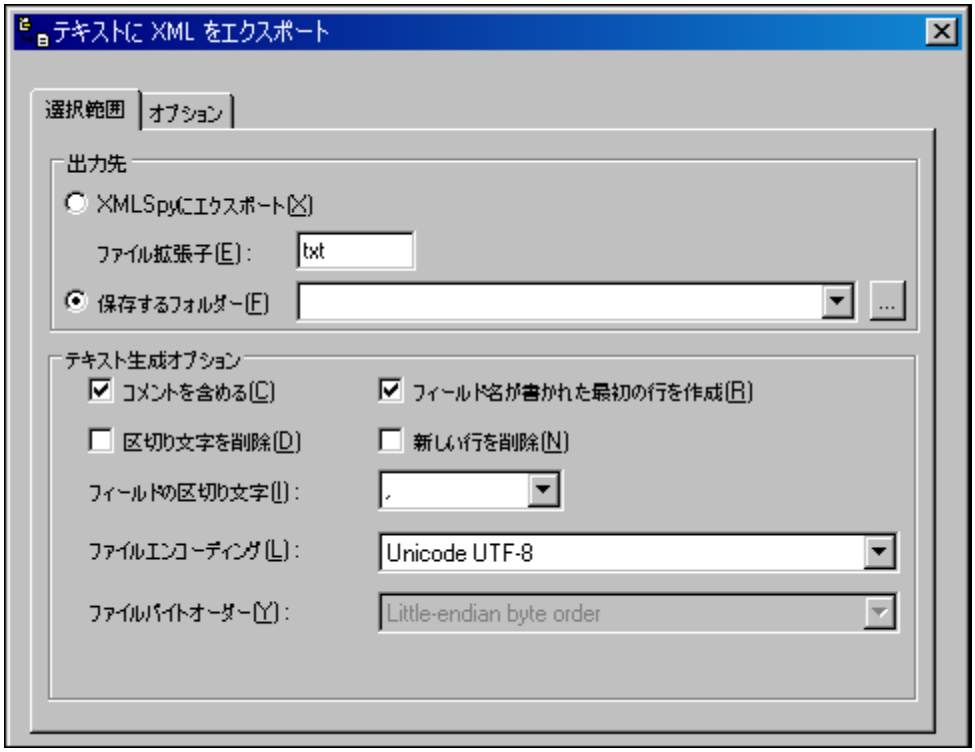
- 上部にある2つのタブ: (i) 選択範囲ならびに (ii) エクスポートオプション
- 下部にあるプレビューウィンドウ

ダイアログにて目的のオプションを選択した後、**エクスポート** ボタンをクリックしてテキストファイルへのエクスポートを行います。

選択範囲

選択範囲タブでは、エクスポートするファイルの出力先と、テキスト生成に関するオプションを指定することができます(以下のスクリーンショットを参照)。

(C) 2015-2021 Altova GmbH



出力先：ファイルをフォルダーへ直接エクスポートします。ファイルの拡張子を指定することができます。ファイル名は、エクスポートするXMLファイル内にある要素の名前により決定されます。他にも、新規ファイルとしてXMLSpy へエクスポートすることもできます。この場合ファイルはGUI にて表示され、後から保存することができます。.

コメントを含める：このオプションを有効にすると、データの取得に使用されるSQL クエリや、データベーステーブル内にあるヘッダーカラムのリストといった、XML ファイル内にあるコメントがエクスポートされます。

フィールド名が書かれた最初の行を作成：有効になっていると、エクスポートされたテーブルにデータベースカラム名が含まれます。

区切り文字を削除：エクスポートされたデータのテキスト値内にある区切り文字を削除します。コンボボックスから削除する区切り文字を指定してください。例えばこのオプションが有効になっており、アポストロフィーが指定されている場合、Ba'ker というXML 値をエクスポートすると、実際にエクスポートされる値はBaker となります。

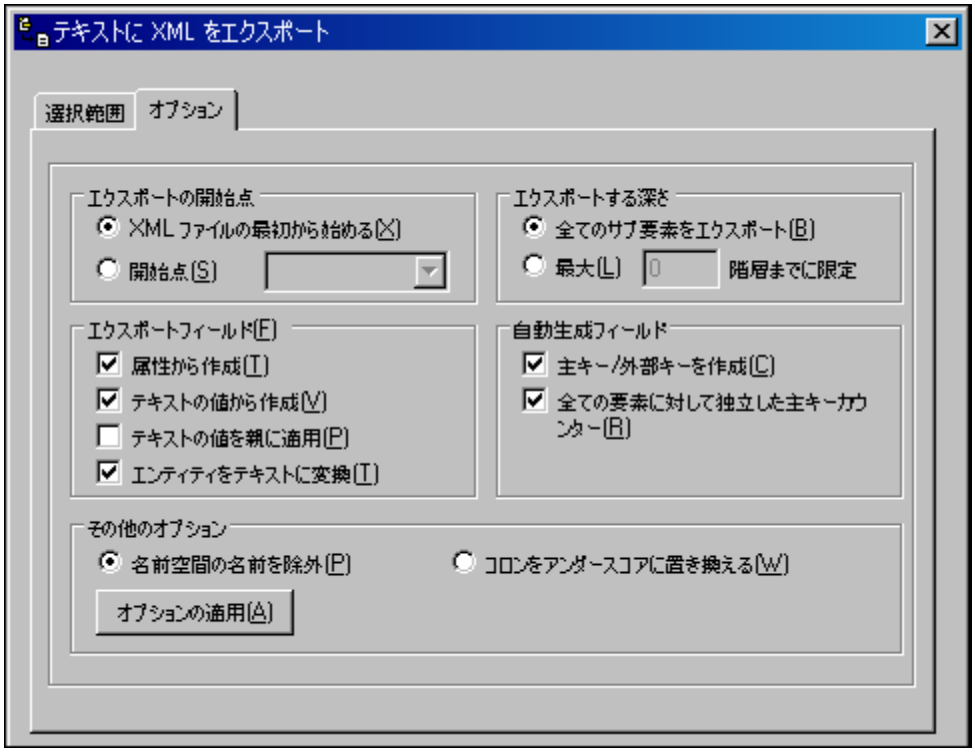
新しい行を削除：エクスポートされるデータから新しい行を削除します。

文字列引用句：このコンボボックスにて指定された引用句が、データの各文字列において使用されます。

バイトオーダー：16ビットまたは32ビットユニコード（UCS-2、UTF-16、またはUCS-4）ファイルをエクスポートする場合、リトルエンディアンとビッグエンディアンからバイトオーダーを選択することができます。

エクスポートオプション

以下に記されるその他のエクスポートオプションは、エクスポートタブにて指定することができます（以下のスクリーンショットを参照）。



エクスポートの開始点: : XML ドキュメント全体をエクスポートするか、現在選択されている要素から始まるデータ構造のみをエクスポートするか選択することができます。エクスポートされるサブレベルの深さについては、*エクスポートする深さ*オプションにて指定することができます。

*エクスポートする深さ*開始点から見て、エクスポートされるサブレベルの数を指定します。

エクスポートフィールド: XML データによって、要素や属性、または要素内のコンテンツだけをエクスポートすることができます。プレビューウィンドウでは、個々の要素に対してエクスポートするかを選択することもできます。

自動生成フィールド :XMLSpy では選択された各要素の型に対して1つの出力ファイルやテーブルが生成されます。主キーや外部キーのペアを自動的に作成することで、データをリレーショナルにリンクしたり、各要素に対して主キーを定義することができます。

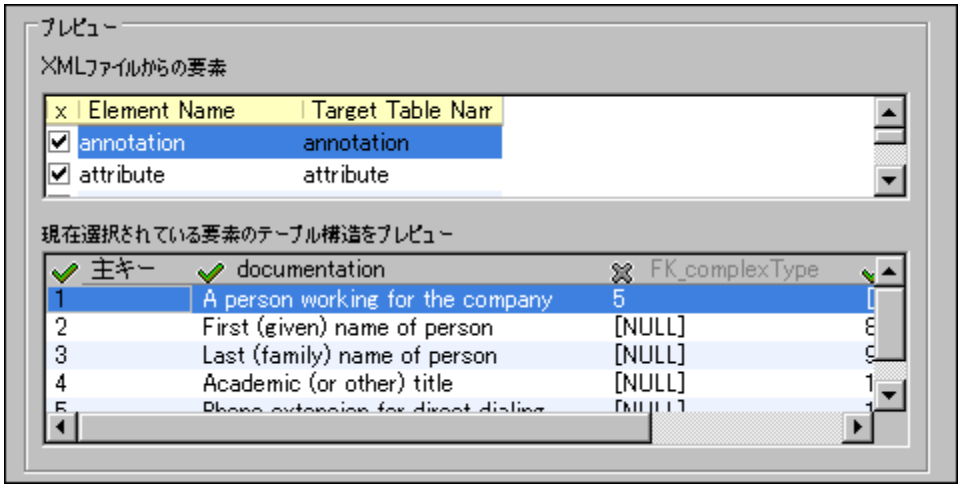
名前空間の名前を削除: コロンをアンダースコアに置き換えるラジオボタンと択一のオプションになります。要素や属性の名前空間プレフィックスを削除するか、名前空間プレフィックスのコロンをアンダースコアに置き換えるか選択することができます。

オプションの適用: オプションが設定されると、このボタンをクリックして、オプションを適用します。プレビューペイン内のプレビューは新しいオプションと共に更新されます。

プレビューウィンドウ

プレビューウィンドウは選択範囲ならびにエクスポートオプションタブ以下に表示されます。

(C) 2015-2021 Altova GmbH



XML ファイルからの要素ペインには、エクスポートされるノードの名前と、生成されるファイルにおける名前が表示されます。エクスポートされるノードを選択/選択解除することができます。要素が選択されていると、その構造が下部にあるペインにて表示されます。このペインにて、カラムの左をクリックすると、そのカラムをエクスポートするかどうか選択することができます。上のスクリーンショットでは、最後の列（FK_Office）がオフに切り替えられています。

24.11.8 データベースへエクスポート

「**変換 | データベースへエクスポート**」コマンドにより、XML データをデータベースへエクスポートします。このコマンドをクリックすることにより表示される接続ウィザードにより、更新を行うデータベースへの接続を行うことができます。接続が確立されると、データベースにXML データをエクスポートダイアログが表示されます。ダイアログは2つのパートから構成されます（それぞれ別々にスクリーンショットを以下に示します）。

- 上部にある2つのタブ: (i) 選択範囲ならびに (ii) エクスポートオプション
- 下部にあるプレビューウィンドウ

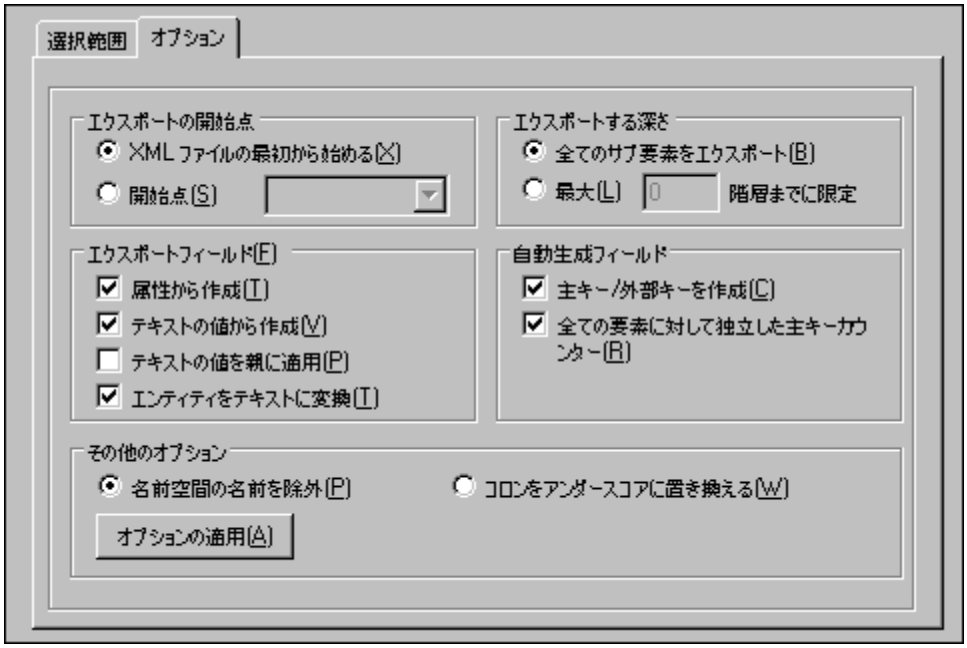
ダイアログにて目的のオプションを選択すると、**エクスポート ボタン**をクリックしてテキストファイルへのエクスポートを行います。

選択範囲

選択範囲タブでは、出力先のデータベースとテーブル生成オプションを選択することができます。出力先フィールドではデータベースへの接続を選択します。データを新たなテーブルとして作成するのか、既存のテーブルを更新するのか、または既存テーブルの更新を試み、それができない場合に新たなテーブルを作成するか選択することができます。エラーの回数による停止アクションセットしたり、SQL スクリプトのログを取るオプションが用意されています。

エクスポートオプション

以下に記されるエクスポートオプションは、エクスポートタブにて指定することができます（以下のスクリーンショットを参照）。



エクスポートの開始点: : XML ドキュメント全体をエクスポートするか、現在選択されている要素から始まるデータ構造のみをエクスポートするか選択することができます。エクスポートされるサブレベルの深さについては、*エクスポートする深さ*オプションにて指定することができます。

*エクスポートする深さ*開始点から見て、エクスポートされるサブレベルの数を指定します。

エクスポートフィールド: XML データによって、要素や属性、または要素内のコンテンツだけをエクスポートすることができます。プレビューウィンドウでは、個々の要素に対してエクスポートするかを選択することもできます。

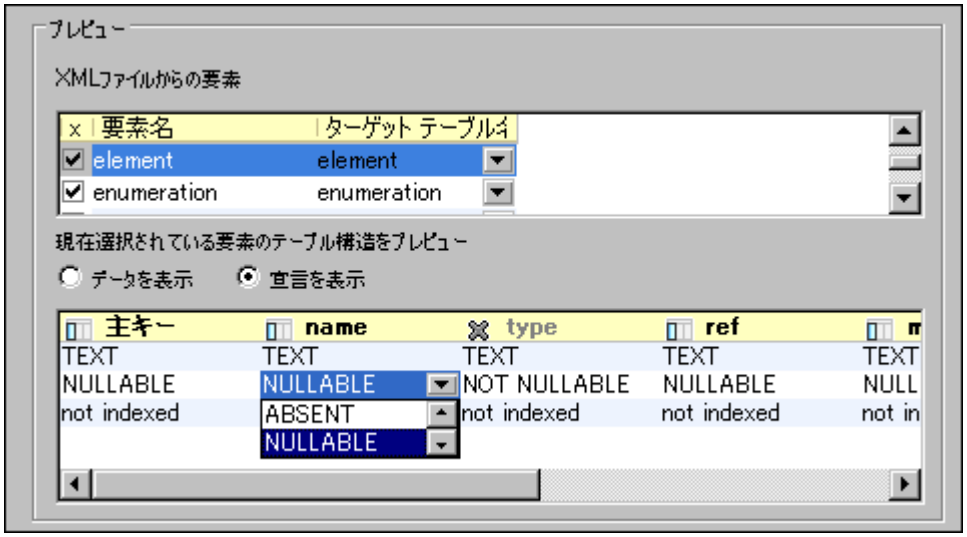
自動生成フィールド :XMLSpy では選択された各要素の型に対して1つの出力ファイルやテーブルが生成されます。主キーや外部キーのペアを自動的に作成することで、データをリレーショナルにリンクしたり、各要素に対して主キーを定義することができます。

名前空間の名前を削除: コロンをアンダースコアに置き換えるラジオボタンと択一のオプションになります。要素や属性の名前空間プレフィックスを削除するか、名前空間プレフィックスのコロンをアンダースコアに置き換えるか選択することができます。

オプションの適用: オプションが設定されると、このボタンをクリックして、オプションを適用します。プレビューペイン内のプレビューは新しいオプションと共に更新されます。

プレビューウィンドウ

プレビューウィンドウでは選択範囲ならびにエクスポートオプションタブ以下に表示されます(以下のスクリーンショットを参照)。



XML ファイルからの要素ペインには、エクスポートされるノードの名前と、生成されるファイルにおける名前が表示されます。エクスポートされるノードを選択/選択解除することができます。要素が選択されていると、その構造が下部にあるペインにて表示されます。このペインにて、カラムの左をクリックすると、そのカラムをエクスポートするかどうか選択することができます。上のスクリーンショットでは、左から数えて3つめのカラムが無効になっており、エクスポートの対象から外されます。このプレビューは以下のプレビューの間を切り替えることができます: (i) 生成された構成内のデータ (*データの表示*); または (ii) 生成された構成内の各列の定義 (*定義の表示*)。上のスクリーンショットは、列の定義を表示しています。

2番目のペインでは、列名の左をクリックすると、列を以下の4つの設定でサイクルします: (i) テーブル構成内に含む (ii) 一意の制約 (iii) 主キーの制約 (iv) テーブル構成空除外 上のスクリーンショットでは、`Location` 列には一意の制約があり、`Fax` 列は、テーブル構成から除外されます。他の全ての列は、テーブル構成内に含まれています。

要素のテーブル構成フィールドの定義を表示する場合 (*定義の表示*)、定義は定義を選択して、定義のコンボボックスからオプションを選択することにより編集することができます (*上のスクリーンショット参照*)。

24.11.9 XML インスタンス を JSON に変換する (JSON から XML インスタンスに変換する)

アクティブなドキュメントがXMLドキュメントの場合、このコマンドは、JSON ドキュメントを生成します。アクティブなドキュメントがJSON ドキュメントの場合、コマンドは、XML ドキュメントを生成します。生成されたドキュメントは、新規のウィンドウで開かれ、希望する場所で保存することができます。 変換オプションについては下で説明されています。に関する詳しい情報は、XMLSpy 内でのJSON とJSON 編集へのサポートに関しては、以下のセクションを参照してください: JSON とJSON スキーマ 462。

変換のサンプル

ソースXML ドキュメントのサンプルは以下のとおりです。JSON ドキュメントは、「 XML インスタンスを JSON に変換する」(JSON から XML インスタンスに変換する) コマンドにより生成されます。

XML ドキュメント

```
<?xml version="1.0" encoding="UTF-8"?>
<Person first="Jim" last="James">
  <Address>
```

```
    <street>4 New Street</street>
    <city>New York</city>
    <state>NY</state>
    <code>10123</code>
  </Address>
  <Tel type="home">
      123 123-1234
  </Tel>
  <Tel type="office">
      123 987-9876
  </Tel>
</Person>
```

JSON ドキュメント

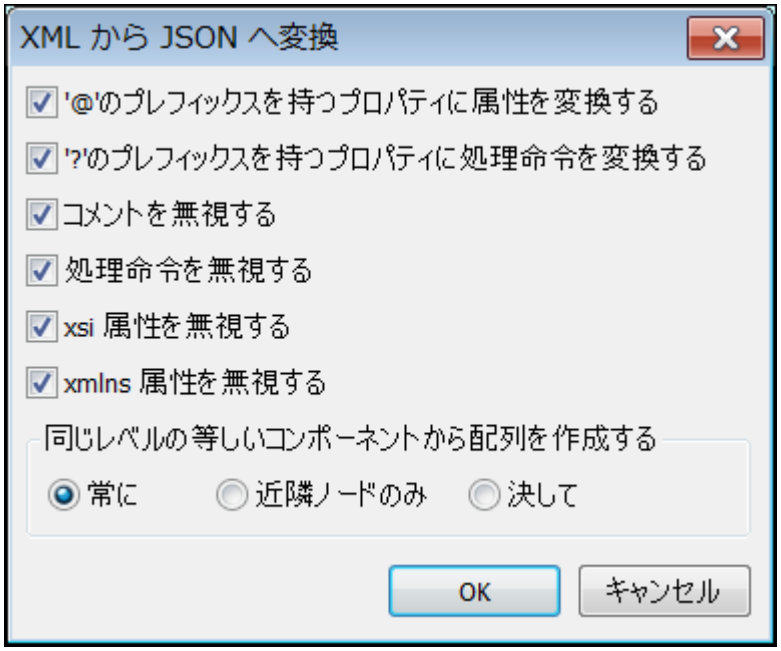
```
{
       "XML": {
              "version": 1.0,
              "encoding": "UTF-8"
       },
       "Person": {
              "first": "Jim",
              "last": "James",
              "Address": {
                     "street": "4 New Street",
                     "city": "New York",
                     "state": "NY",
                     "code": 10123
              },
              "Tel": [ { "type": "home",
                     "Text": "\r      123 123-1234\r  "}, { "type": "office",
                     "Text": "\r      123 987-9876\r  "} ]
       }
}
```

JSON ドキュメントを XML に変換するには、JSON ドキュメントをアクティブにし、「XML インスタンスを JSON に変換する」（JSON から XML インスタンスに変換する）コマンドをクリックします。

XML から JSON への変換オプション

「XML インスタンスを JSON に変換する」（JSON から XML インスタンスに変換する）コマンドをクリックすると、XML インスタンスドキュメントを JSON インスタンスドキュメントに変換する XML から JSON への変換 ダイアログ（*下のスクリーンショット*）が表示されます。変換オプションを設定し、「OK」をクリックします。JSON インスタンスドキュメントが XML インスタンスから生成され、生成された JSON ドキュメントは、新しいウィンドウでひらかれます。

(C) 2015-2021 Altova GmbH



最初の2つのオプションは、同じレベルでの要素の競合を回避するためにプレフィックスがJSON プロパティ名 に追加されるかを定義します。下の2つのリストが上記を説明しています。XML *属性* `somenode` は、JSON プロパティ `@somenode` に変換されています。このように、(XML 要素 `somenode` から作成された) JSON プロパティ `somenode` との競合が回避されます。

XML インスタンス

```
<root somenode="value">
    <somenode>content</somenode>
</root>
```

JSON instance

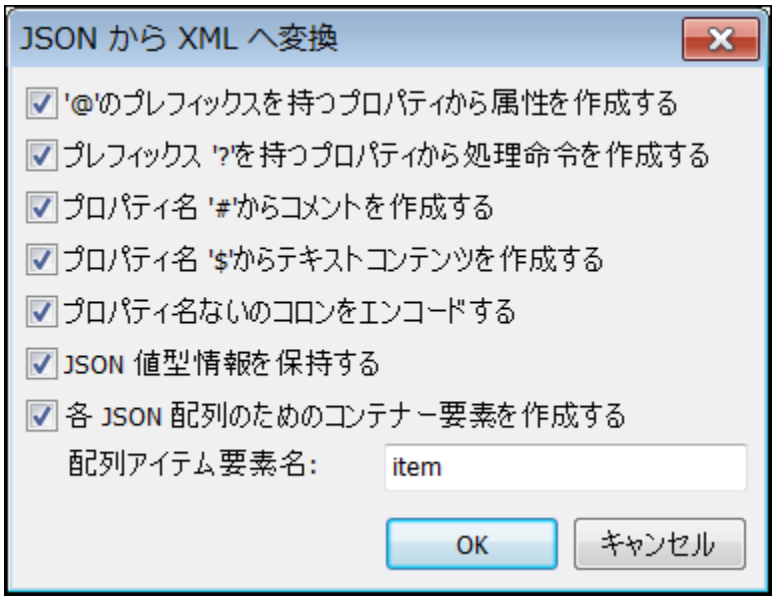
```
{
    "root": {
        "@somenode": "value",
        "somenode": "content"
    }
}
```

次のオプションにより、特定のXML ノードの型が変換されるかを指定することができます。XML コメントが含まれている場合は、`"#"` という名前が与えられます。(複合型コンテンツを持つ要素内で通常発生する)テキストノードは `"$"` という名前を与えられます。XML ノードが名前空間プレフィックスを持つ場合、対応するJSON の名前は、この名前空間プレフィックスを使用して作成されます。同じ名前を持つ要素が、同じレベルに存在する場合、等価 コンポーネントと考えられます。同様に、同じレベルに存在するノード、処理命令、とテキスト() は、等価 コンポーネントと考えられます。等価 コンポーネントが同じレベルに存在すると、配列を作成するか否かを選択することができます。すべての等価 コンポーネント、近隣する等価コンポーネントから配列を作成するか、または、配列を作成しない オプションを選択することができます。

XML スキーマ型情報の使用 オプションによりソースノードのXML スキーマ型をベースに行うことができます。例えば、ノードがスキーマ内で型 `xs:string` として定義されている場合、ターゲット JSON オブジェクトプロパティはJSON 型 `string` になり、引用符で囲まれます。これはとても役に立ちます。例えば、ソースノード内で文字列として保管されている数値のJSON への変換は型に従い行われます。.

JSON からXML への変換オプション

「XML インスタンスを JSON に変換する」(JSON から XML インスタンスに変換する) コマンドをクリックすると、JSON インスタンスドキュメントをXML インスタンスドキュメントに変換する「JSON からXML に変換」ダイアログ が表示されます(*下のスクリーンショット*)。変換オプションを設定し、「OK」をクリックします。XML インスタンスドキュメントがJSON インスタンスから生成され、生成されたXMLドキュメントは、新しいウィンドウで開かれます。



以下の点に注意してください:

- JSON オブジェクトプロパティは、XML 要素に変換されます。ダイアログ内の最初のオプションによりプロパティの型が作成されるかを選択することができます。
- *プロパティ名内のエンコードコロン:* 選択されると、JSON の名前内のコロンはエンコードされ、コロンとして作成されません。選択されていない場合、コロンはそのままで保持されます。
- *JSON 値の情報の保持:* 選択されると、プロパティのJSON 型 情報は、対応する要素の属性-値ペアとして作成されます。
- *JSON 配列のためにコンテナー要素を作成する:* XML ドキュメント内のコンテナー要素がJSON 配列オブジェクトの名前として与えられます。JSON 配列のアイテムは、このコンテナー内のXML 要素として作成されます。*配列アイテム要素名* テキストボックス内で指定される名前がそれぞれに与えられます。

24.11.10 XML スキーマ を JSON スキーマに変換する(JSON スキーマから XML スキーマに変換する)

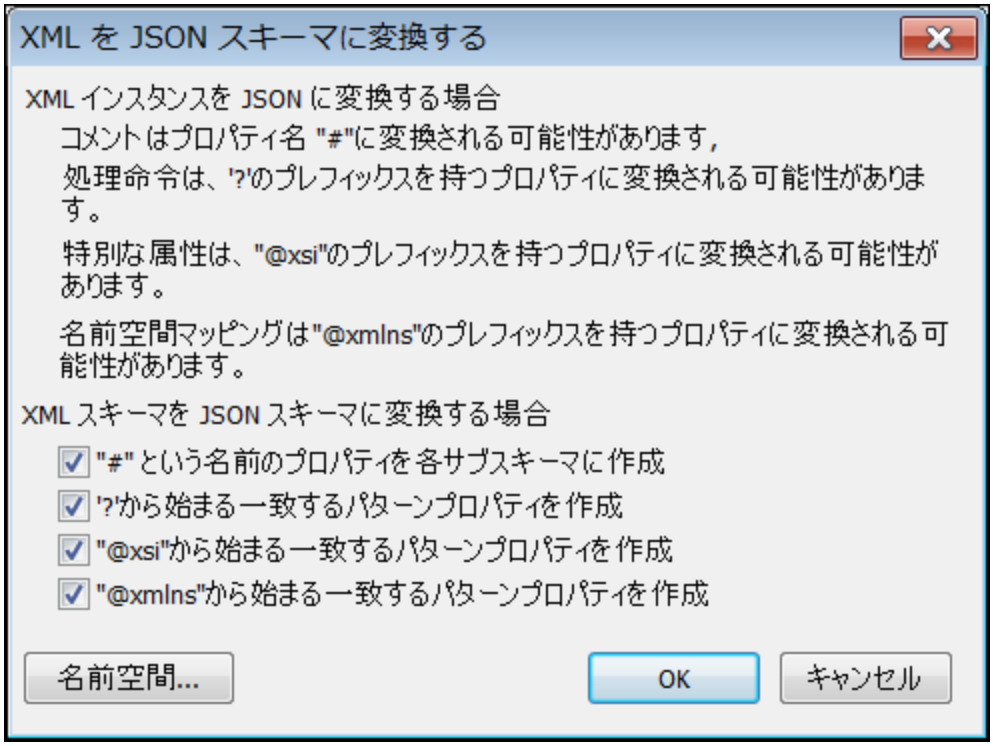
アクティブなドキュメントがXML スキーマの場合、このコマンドは、JSON スキーマドキュメントをXML スキーマから生成します。アクティブなドキュメントがJSON スキーマの場合、コマンドは、XML スキーマをJSON スキーマから生成します。生成されたドキュメント は、新規のウィンドウで開かれ、希望する場所に保存することができます。変換オプションについては下で説明されています。XMLSpy 内のJSON とJSON 編集のためのサポートに関する詳しい情報は、以下のセクションを参照してください:JSON とJSON スキーマ 462。

XML スキーマをJSON スキーマに変換するオプション

「XML スキーマをJSON スキーマに変換する」(JSON スキーマからXML スキーマに変換する) コマンドをクリックすると、XML スキーマドキュメントをJSON スキーマに変換する、「XML スキーマをJSON スキーマに変換する」ダイアログが表示されます(*下のスクリーンショッ*

ト)。必要とするJSON スキーマ バージョン[506] と変更オプションを選択します。「OK」をクリックします。JSON スキーマは、XML スキーマから生成され、生成されたドキュメントは、新しいウィンドウでひらかれます。

一般的な変換の戦略方法は以下のとおりです: (i)(文字列番号などの) XML スキーマ単純型はJSON スキーマ単純型にマップされています(ii) XML スキーマ複雑型はJSON オブジェクトにマップされています。

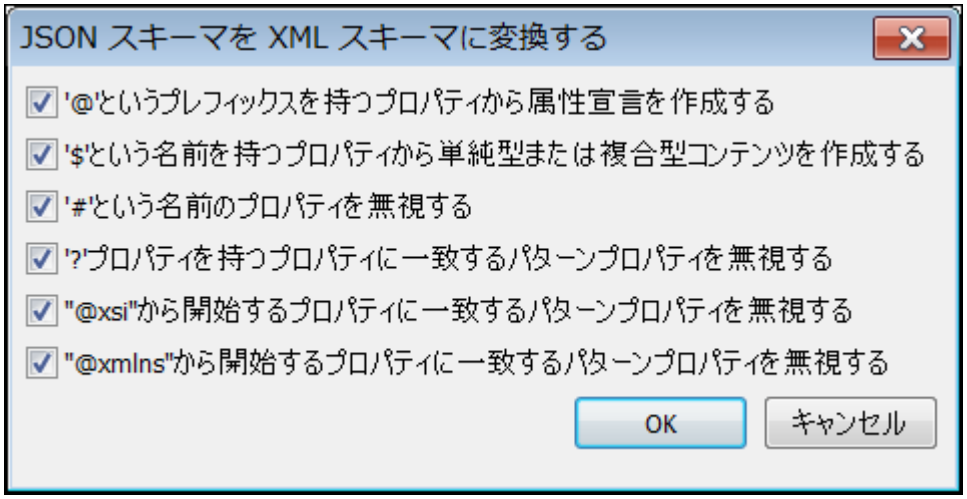


ダイアログの上の部分では、特定のXML スキーマ コンポーネントがどのように変換されるかに関する情報が提供されます。ダイアログの下の部分は、次のオプションを提供します:

- *各サブスキーマ内に"#" という名前のプロパティを作成する*: 選択されると、JSON スキーマ定義内にこの名前を持つプロパティが作成されます。
- *'?'、"@xsi"、"@xmlns"のプレフィックスを持つプロパティに一致するパターンプロパティを作成する*: それぞれのプレフィックスのために、これらのプレフィックスを持つプロパティに一致するパターンプロパティを指定します。パターンプロパティに関する詳しい情報は、以下のセクションを参照してください:JSON オブジェクトとプロパティ[515]。
- *maxOccurs > 1 を使用してパーティクルのための配列を常に作成する*: XML スキーマ内では、パーティクルは複合型コンテンツモデルの要素です。複数回発生する場合、パーティクルはJSON スキーマ内の配列として定義されます。それ以外の場合、JSON オブジェクトのプロパティとして定義されます。
- *ピュアオブジェクトを単純型コンテンツを持つ複合型のために作成する*: 単純型コンテンツを持つXML スキーマの複合型は属性とテキストコンテンツであり子要素ではありません。If the *ピュアオブジェクトの作成* オプションが選択されると、複合型はJSON オブジェクトに変換されます。型の属性はプロパティ名が@プレフィックスを持つJSON オブジェクトのプロパティに変換されます 。型のテキストコンテンツのために$ という名前のプロパティが生成されます。*ピュアオブジェクトの作成* オプションが選択されていない場合、複合型は他のオブジェクトを含むオブジェクトおよび文字列と番号などのJSON 単純型に変換されます。

JSON スキーマをXML スキーマに変換するオプション

「XML スキーマをJSON スキーマに変換する」(JSON スキーマからXML スキーマに変換する) コマンドをクリックすると、JSON スキーマドキュメントをXML スキーマに変換する、「JSON スキーマからXML スキーマに変換する」ダイアログが表示されます(*下のスクリーンショット*)。変換オプションを設定し、「OK」をクリックします。XML スキーマがJSON スキーマから生成され、生成されたドキュメントは、新しいウィンドウでひらかれます。

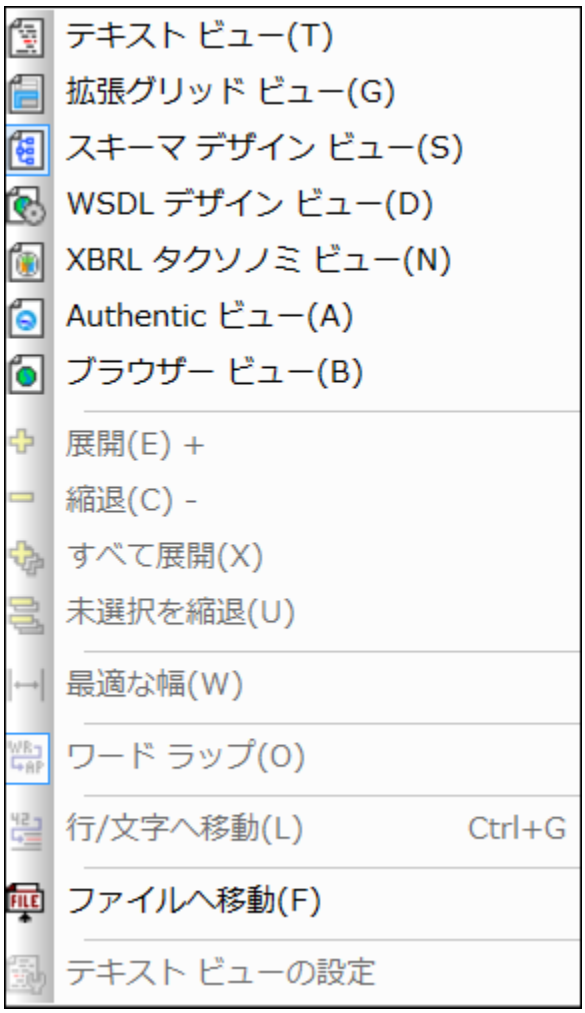


次のオプションから選択することができます:

- `'@'` と `'$'` から開始するJSON プロパティ名が作成されるかどうか。それぞれが属性ノードとテキストノードとして作成されます。
- `'#'` という名前のプロパティがXML コメントノードとして作成されるか。
- `'?'` プレフィックスを持つプロパティに一致するパターンプロパティで `"@xsi"` および/または `"@xmlns"` が無視されるか。無視されない場合、`'?'` プレフィックスを持つプロパティでは、`"@xsi"` と `"@xmlns"` がそれぞれ処理命令 `xsi:` プレフィックス属性 と `xmlns:` プレフィックス属性に変換されます。

24.12 表示メニュー

表示 メニューでは（*下のスクリーンショット*）では、アクティブな メインウィンドウ[104] の表示設定を行い、ドキュメントの表示方法を変更することができます。



このセクションでは、「**表示**」メニューから使用することのできるコマンドを説明します。

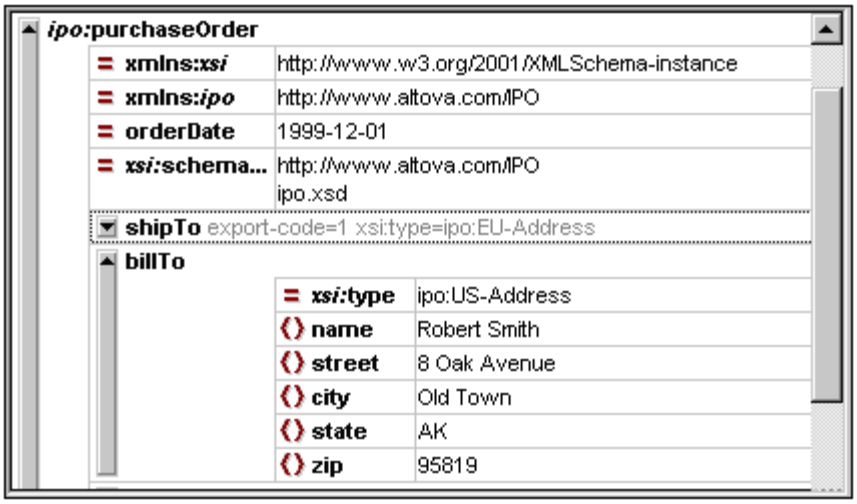
24.12.1 テキストビュー

このコマンドにより、現在使用されているドキュメントのビューを テキストビュー[122] へ切り替えます。テキストビューではテキスト形式でドキュメントの編集を行うことができ、数々のテキスト編集機能がサポートされます。テキストビューに関する詳細は、このドキュメントの テキストビュー[122] セクションを参照ください。

メモ: オプションダイアログ(「ツール| オプション[1014]」)にある様々なタブを使用することで、テキストビューの設定を変更することができます。

24.12.2 拡張グリッドビュー

このコマンドにより、現在のドキュメント表示がグリッドビュー[141] に切り替わります。それまで使用されていたビューがテキストビュー[122]の場合、ドキュメントの整形式が自動的にチェックされます。



24.12.3 スキーマデザインビュー

このコマンドにより、現在のドキュメント表示がスキーマデザインビューに切り替わります。ビューの切り替えは、現在選択されているドキュメントがXML スキーマドキュメントの場合だけ行うことができます。このビューに関する詳細は、このドキュメンテーションのスキーマビュー[151] セクションを参照ください。

24.12.4 Authentic View

このコマンドは、現在のドキュメントをAuthentic View[415] に切り替えます。

Authentic View を使用するとAltova のStyleVision アプリケーションにより作成されたStyleVision Power Stylesheet テンプレートをベースにXMLドキュメントを編集することができます。これらのテンプレート（StyleVision スタイルシート または SPS ファイル）は、XMLドキュメントをグラフィカルな フォーマットで表示し、XMLドキュメントを（マークアップを使用下テキストフォーマット編集するより先）簡単に編集することができます。

XMLドキュメントがSPS ファイル（Authentic | StyleVision スタイルシートの割り当て[903]）と関連付けられている場合、XMLドキュメントをAuthentic View で閲覧することができます。Authentic View 内でSPS ファイルを新規の空のテンプレートとして以下の方法で開くことができます:

- 「ファイル | 新規作成」コマンドを選択して、StyleVision スタイルシートの選択 ボタンをクリックします。
- 「Authentic | 新規ドキュメント」コマンドを選択して、SPS ファイルを参照します。

詳細に関しては、Authentic View[401] とStyleVision のドキュメントを参照してください。

24.12.5 ブラウザービュー

このコマンドにより、現在のドキュメントビューをブラウザービュー[224] へ切り替えます。このビューではXML が利用可能なブラウザーを使用して、CSS やXSL スタイルシートにより得られた情報によるXMLドキュメントのレンダリングを行います。

オプションダイアログのファイルタブ[1015] にて保存時に検証を行うよう設定している場合、ブラウザビューへの切り替えが行われる際に、ドキュメントの検証が行われます。オプションダイアログは、「ツール | オプション」メニューコマンドにより開くことができます。詳細に関しては、このドキュメントのブラウザービュー[224] のセクションを参照してください。

24.12.6 展開

このコマンド（*ショートカット:* 数字パッド '+'）はグリッドビューにて使用することができます。このコマンドにより、選択されている要素が1段階展開されます。展開が行われると、その要素と子要素が全て選択された状態になります。この機能により、"+" キーを繰り返し押下することで、大きな要素の展開を行うことができます。

24.12.7 縮退

このコマンド（*ショートカット:* 数字パッド '-'）はグリッドビューにて使用することができます。このコマンドにより、選択されている要素が1段階縮退されます。
各要素の左に配置されているグレーのバーをクリックすることでも、要素の展開や縮退を行うことができます。

24.12.8 全て展開

このコマンド（*ショートカット*: * または数字パッド上で x ）はグリッドビューにて使用することができます。テキストビューの縮退マージンがアクティブな場合テキストビューにて使用することもできます。選択された要素のすべての子孫ノードを展開します。

24.12.9 未選択を縮退

このコマンド（ショートカット: **Ctrl** + 数字パッド **'-'**）はグリッドビューにて使用することができます。選択されている要素の周りにある要素が縮退され、その要素とその子孫だけにフォーカスすることができます。目的の要素を選択して、このコマンドを選択することで、その他全ての（選択されていない）要素が縮退されます。

24.12.10 最適な幅

このコマンドにより、セルに含まれている最も長い文字列が1行に収まるように全カラムの幅が調整されます。オプションダイアログ（「**ツール | オプション**」）のビューセクションでは、最適幅の最大サイズを指定することができます。最適幅は、表示されているカラムのセルにより計算されているということに留意してください。この機能により、個々の要素が展開または縮退された時に、最適な表示を行うことができるようになります。

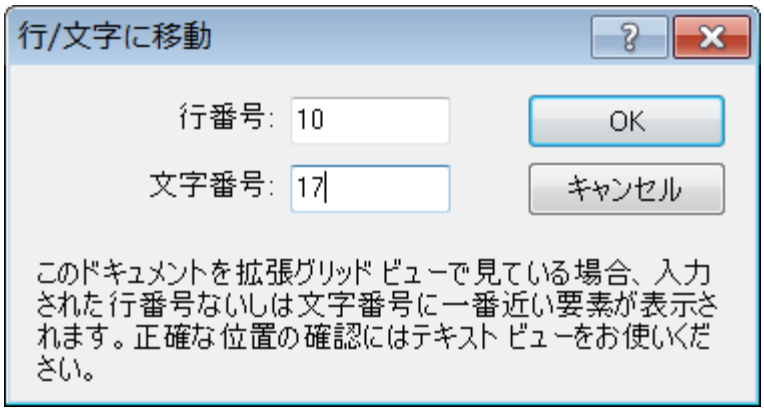
24.12.11 ワードラップ

このコマンドにより、「**テキストビュー**」におけるワードラップ（右端折り返し）を有効化/無効化することができます。ワードラップがオンに切り替えられると、テキストはウィンドウの端で折り返されます。

(C) 2015-2021 Altova GmbH

24.12.12 行/文字へ移動

このコマンド（*ショートカット:* **Ctrl+g**）テキストビューとグリッドビュー利用することができます。移動先の文字と数字を入力することのできるダイアログ（*下のスクリーンショット*）が表示されます。テキストビューでは、入力された位置にカーソルが移動します。グリッドビューでは、特定の行番号および文字の位置に最も近いノードがハイライトされます。



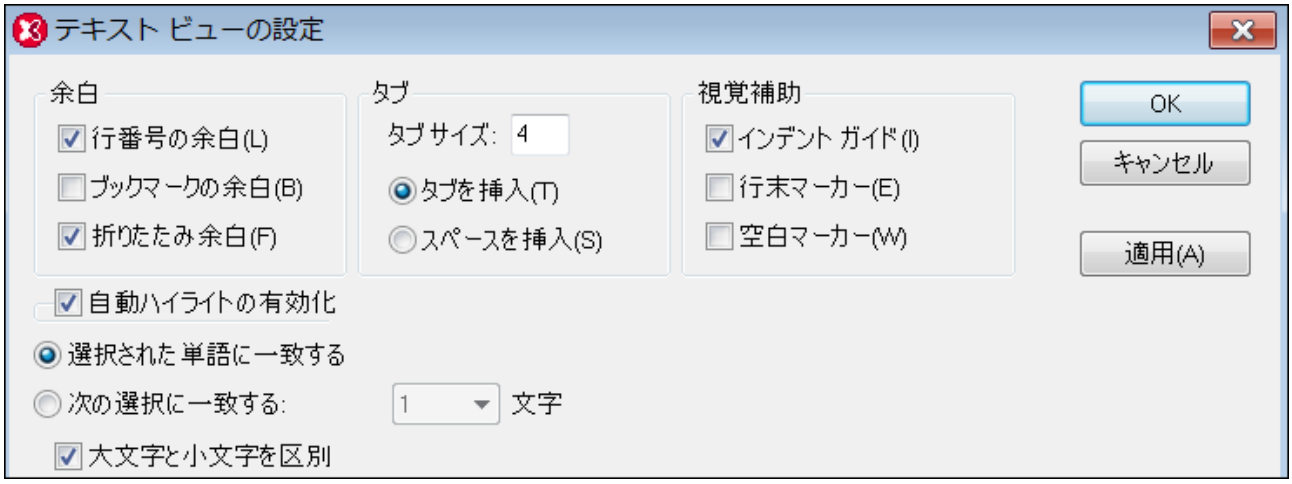
このコマンドは、テキストビューならびにグリッドビューの両方で使用することができますが、グリッドビューでは、指定された文字位置に最も近い要素をハイライトすることで、大まかな位置が示されます。

24.12.13 ファイルへ移動

このコマンドはテキストビューとグリッドビュー内で有効化することができます。（テキストビュー内で）カーソルがファイルを参照するテキスト内、または、（グリッドビュー内で）ファイルを参照するテキストを含むノード内にカーソルが置かれると、参照されるドキュメントが開かれます。現在編集中のファイル内から、参照されているドキュメントが開かれます。

24.12.14 テキストビュー設定

テキストビュー設定コマンドにより、テキストビュー設定ダイアログが表示され、テキストビューの設定を行うことができます（以下のスクリーンショットを参照）。テキストツールバーに配置されているショートカットアイコン を使用することでも、このコマンドを使用することができます。



余白

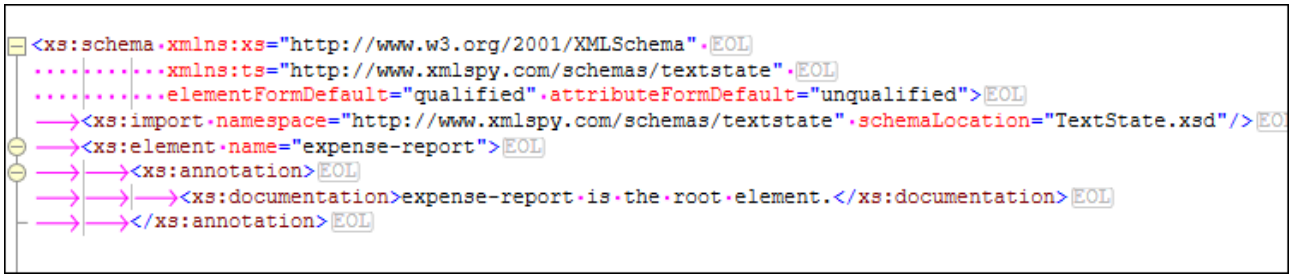
余白ペインでは、行番号、ブックマーク、そして折りたたみ余白を有効/無効にすることができます。これらはそれぞれ異なる種類の余白で、行番号、ブックマーク、ソースの折りたたみ(展開や縮退を行うためのアイコン)がそれぞれ表示されます。これら設定により、テキストビューにて余白が表示されるかが決定されます。ブックマークコマンドは編集メニュー内に配置されています。ノードの展開や縮退を行うには、折りたたみ余白が有効になっていなければなりません。

タブ

タブペインでは、タブの大きさを指定することができます。タブを挿入するかスペースを挿入するか決定するラジオボタンにより、インデントを使った整形がオプションダイアログのビュータブ(1025 ツール | オプション 1025)1025内で有効になっている際にタブを使用するのか、スペースによるインデントを行うかが決定されます。

視覚補助

視覚補助ペインには、インデントガイド(テキストのインデントを表示するための縦方向に走る破線)、改行マーカー、そして(下のスクリーンショットでピンク色で表示されている)空白マーカーを表示するための設定が含まれます。タブは矢印にて、空白スペースは点線で示されています。空白スペースマーカーは、オプションダイアログのテキストビューオプション1029内でカスタム化することができます。



ハイライトの有効化

ハイライトが有効化されていると、テキストビュー内の選択の発生する箇所全てがハイライトされます。このペイン内のオプションにより選択項目が構成されます。選択は、大文字と小文字を区別しつつ用語、または、文字数に定義することができます。文字の選択の場合は、一

致する文字の最小数を指定することができます（例えば、2つまたは複数の文字）。テキストビュー内では、選択に一致する文字のシーケンスの発生すべてがハイライトされます。用語検索の場合、引用符無しの要素名、属性名、属性値、と要素タグの角ブラケットが区切られている文字と考えられます。

キーマップ

キーマップコは、XMLSpy のショートカットと関連するコマンドのリストが表示されます。

24.13 ブラウザーメニュー

「ブラウザー」メニューのコマンドは、ブラウザービュー 224 で有効になります。「戻る」と「進む」コマンドはしかしながら、前回使用されたコマンドに移動することのできるスキーマビューでも有効化されることができます。

24.13.1 戻る

戻る コマンド（*ショートカット:* Alt + 左矢印）は、ブラウザービューとスキーマビューで有効化されます。

ブラウザービューにて、「戻る」コマンドにより、その前に表示されたページが表示されます。バックスペースキーを押下することでも同様の操作を行えます。戻るコマンドは、XML ドキュメントのリンクをクリックした後に、再度 XML ドキュメントへ戻るために使用することができます。

スキーマビューでは、戻るコマンドにより、その前に閲覧していたコンポーネントが表示されます。Alt + 左矢印がショートカットキーになります。戻るコマンドにより、最大500回前の閲覧箇所まで戻ることができます。

24.13.2 進む

進む コマンド（*ショートカット:* Alt + 右矢印）は、ブラウザービュー内で有効化されます。「進む」コマンドは、「戻る」コマンドを使用した後に有効になります。このコマンドにより、(i) ブラウザビューにて既に閲覧したページや、(ii) スキーマビューにて既に閲覧したスキーマコンポーネントを閲覧することができます。

24.13.3 中止

中止コマンドにより、ブラウザービューで行われているドキュメントの読み取りを中止します。スピードが遅いインターネット回線などにより外部ファイルや画像などがダウンロードされており、その処理を停止したい場合にこのコマンドを使用することができます。

24.13.4 最新の状態に更新

最新の状態に更新 (F5) コマンドにより、ブラウザービューのドキュメントや、CSS、XSL スタイルシート、DTD といった関連するドキュメントが再ロードされます。

24.13.5 フォント

フォントコマンドにより、XML ドキュメント内にあるテキストのレンダリングに使用されるデフォルトのフォントサイズを選択することができます。殆どのブラウザーにおける文字のサイズコマンドに似た動作をします。

24.13.6 別のウィンドウ

別のウィンドウで開くコマンドにより、ブラウザービューが別のウィンドウにて開かれ、他のビューと隣合わせで表示することができるようになります。ブラウ別のウィンドウでは、ブラウザービューはドキュメントの編集ビューと左右に表示することができます。

ブラウザービューが別のウィンドウで表示されていると、編集ビューにて **F5** を押下することで、対応するブラウザービューの内容も更新されます。ウィンドウを元のインターフェースに統合するには、アクティブなウィンドウの右上にある最大化ボタンをクリックしてください。

24.14 ツールメニュー

ツールメニューを使用して以下を行うことができます:

- XML ドキュメントのスペル[974] チェックをチェックします。
- XMLSpy のスクリプト環境[1056] にアクセスします。フォーム、マクロ、イベントハンドラーを作成、管理、および保管することができます。
- 現在割り当てられているマクロをビュー[983]します。
- 差異をチェックするために2つのファイルを比較する。
- 差異をチェックするために2つのフォルダーを比較する。
- 外部アプリケーションを使用するカスタマイズされたコマンドにアクセスします。これらのコマンドは、カスタマイズ ダイアログのツールタブ内で作成することができます[1002]。
- グローバルリソースの定義[993]
- XMLSpy 内のグローバルリソースのためにアクティブな構成を変更します[994]。
- XML と XBRL の検証と RaptorXML 検証オプションを構成するために[996] RaptorXML Servers を追加します[995]。
- アクティブな構成として Raptor Server 構成を選択します[998]
- XMLSpy の自身のバージョンをカスタマイズします[998]: 自身のツールバー、キーボードショートカット、メニューとマクロを定義します。
- グローバルな XMLSpy 設定[1014]を定義します。

24.14.1 スペリング

内蔵の言語辞書を搭載した XMLSpy のスペルチェッカーは、テキストビュー、グリッドビュー、および Authentic View. で使用することができます。その他のビューで編集したドキュメントに対してスペルチェックを行いたい場合は、テキストビューまたはグリッドビューへ切り替えた後にスペルチェックを起動してください。例えば、XML スキーマドキュメントをスキーマビューで編集している場合、テキストビューまたはグリッドビューへの切り替えを行い、スペルチェックを行うことができます。

メモ: Altova ソフトウェアに内蔵されている辞書は、Altova の好みにより選択されたものではありません。MPL、LGPL、または BSD ライセンスといった商用ソフトウェアにおける再配布が認められているライセンス下で利用可能かどうかに依存しております。GPL ライセンスなどの、より強力なライセンス下で配布されています。これらの辞

書はhttps://www.altova.com/ja/dictionaries に配置されているインストーラーからご利用いただくことができます。ライセンスと有用性を基に使用する辞書を選択してください。

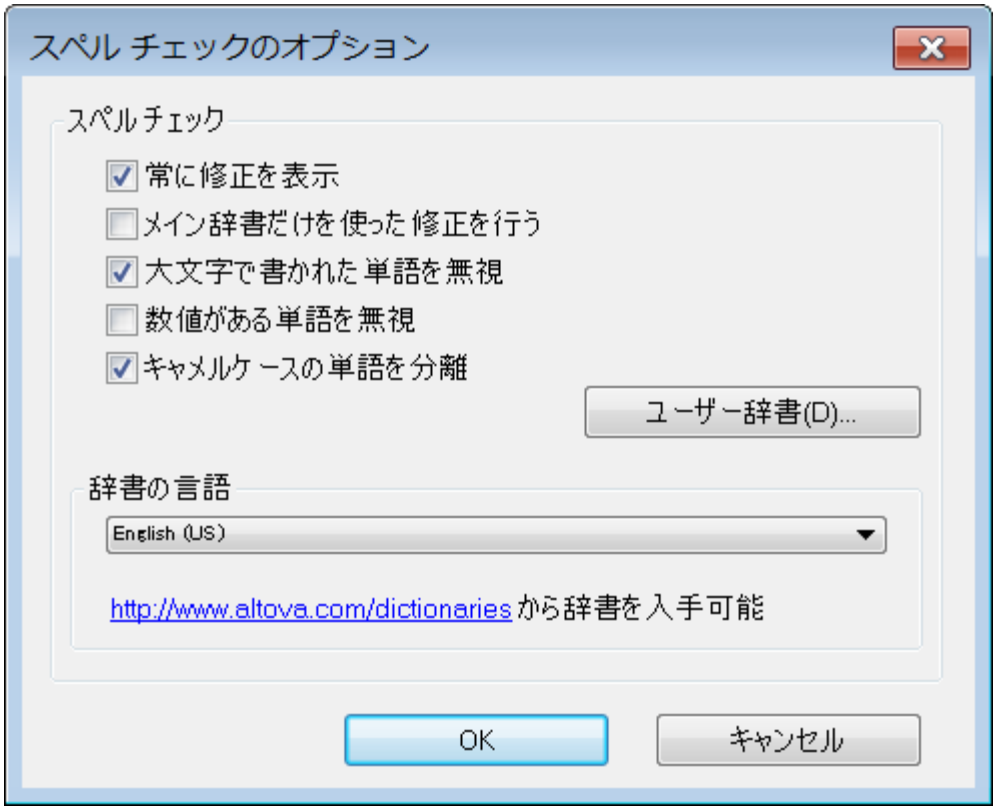
このセクションではスペルチェッカーの使用方法について記述され、以下の3つのサブセクションから構成されます:

- スペルチェッカーの言語を選択 975
- チェックを行う範囲の定義 975
- スペルチェッカーを起動する 976

スペルチェッカーの言語を選択

以下の操作により、スペルチェックを行う言語をセットすることができます:

1. 「ツール| スペルチェックのオプション」メニューコマンドをクリックします。
2. スペルチェックのオプションダイアログにて、辞書の言語コンボボックスのリストからインストールされた辞書を選択します(以下のスクリーンショットを参照)。



3. **OK** をクリックして完了します。

選択された言語の辞書がスペルチェックに使用されます。スペルチェックを行う言語の辞書がまだインストールされていない場合、新たな辞書をダウンロードすることができます。ダウンロードの方法については、スペルチェッカーに辞書を追加 981 のセクションを追加を参照ください。

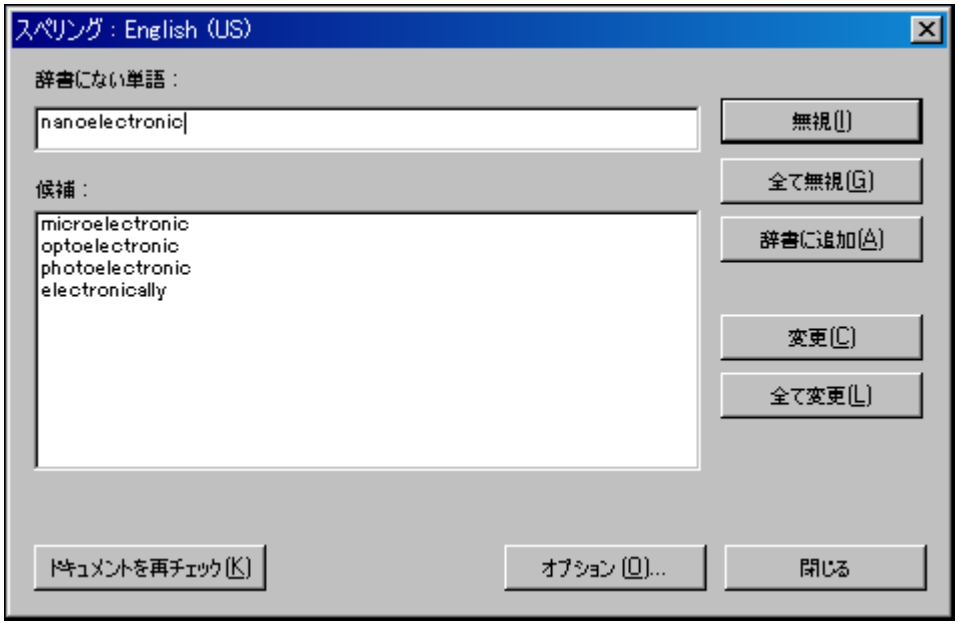
チェックを行う範囲の定義

テキストまたはグリッドビューでスペルチェックが実行される場合、チェックを開始する直前にチェックする範囲を定義することができます。コマンド[ツール| スペリングオプション] を選択して表示される XML スペルチェックのオプションのダイアログ 978 内で目的の範囲を定義します(以下のスクリーンショットを参照)。

チェックを行う要素や属性を選択できるほか、CDATA セクションやコメントをチェックするか指定することができます。スペルチェックのオプション 978 コマンドの記述も参照ください。

スペルチェッカーを実行する

「ツール | スペルチェック」コマンド（Shift + F7）により、現在アクティブになっているXMLドキュメント内における、定義づけされた範囲 975 に対してチェックが自動的に始まります。未知の単語が発見されると、スペリングダイアログが表示されます（以下のスクリーンショットを参照）。未知の単語が無い場合、スペルチェックがドキュメントの最後まで行われます。



スペリングダイアログで使用することのできるオプションを以下に記します:

辞書にない単語

このテキストボックスには、選択された言語辞書やユーザー辞書にて発見できなかった単語が表示されます。以下のオプションが利用可能です:

- テキストボックスに表示されている単語をキーボードを使って編集するか、候補リストから選択することができます。その後変更ボタンをクリックすることで、XMLドキュメント内にある単語が、編集された単語に置き換えられます(候補リストにある単語をダブルクリックすることで、XMLドキュメントの内容が直接置き換わります)。単語が辞書にない単語テキストボックスに表示されると、XMLドキュメント内の対応する箇所がハイライトされ、ドキュメント内の単語を直接編集することもできます。全て変更をクリックすることで、置換処理がXMLドキュメント全体に対して行われます。
- 変更を行わず、現在ハイライトされている箇所のみ、またはドキュメント全体においてスペルチェッカーの警告を無視することができます。
- 単語をユーザー辞書に追加することで、それ以降のチェックにおいてその単語が正しいものであると認識させることができます。

候補

このリストボックスには、未知の単語に近い(言語辞書ならびにユーザー辞書から得られた)単語が表示されます。リストにある単語をダブルクリックすることで、その単語がドキュメント内へ挿入され、スペルチェック処理が続けられます。

一度無視する

このコマンドにより、現在選択されている未知の単語を無視して、ドキュメントのチェックを続けることができます。同じ単語がドキュメント内で再度発見された場合、その単語が再度表示されます。

全て無視

このコマンドにより、現在の未知単語をドキュメント全体において無視することができます。

辞書に追加

このコマンドにより、未知の単語を**ユーザー辞書**へ追加します。ユーザー辞書へはスペルチェックのオプション 981 ダイアログにてアクセス、編集を行うことができます。

変更
このコマンドにより、XML ドキュメント内で現在ハイライトされている単語が、(変更された) *辞書にない単語* テキストボックス内の単語に置き換えられます。

全て変更
このコマンドにより、XML ドキュメント内で現在ハイライトされている単語が、ドキュメント全体で(変更された) *辞書にない単語* テキストボックス内の単語に置き換えられます。

ドキュメントを再チェック
ドキュメントを再チェック ボタンにより、ドキュメントの先頭から再度チェックを行います。.

オプション
現在使用されているビューにより以下のダイアログボックスが表示されます:

- 現在のビューが Authentic View の場合、スペルチェックのオプション[978] ダイアログボックスが表示されます。
- 現在のビューがテキストビューまたはグリッドビューの場合 XML スペルチェックのオプション[978] ダイアログボックスが表示されます。

ダイアログボックスに関する詳細はスペルチェックのオプション[978] セクションを参照ください。

閉じる
スペリングダイアログボックスが閉じられます。

24.14.2 スペルチェックのオプション

「**ツール | スペルチェックのオプション**」コマンドにより スペルチェックのオプション[979] が表示されます現在アクティブになっているビューに従い、「**ツール | スペルチェックのオプション**」コマンドにて、スペルチェックのオプション[979] ダイアログ(スキーマビュー、Authentic ビュー、ブラウザービュー)、または XML スペルチェックのオプション[978] ダイアログ(テキストビュー、グリッドビュー)が表示されます。コンテキストダイアログにある「**スペリングオプション**」ボタンにて、スペルチェックのオプションダイアログを表示することができます。

これら2つのダイアログでは、以下に記される様々な設定を行うことができます:

- スペルチェックのコンテキスト[978]
- スペルチェックのオプション[979]
- スペルチェッカーに辞書を追加[981]
- ユーザー辞書と作業[981]

XML スペルチェックのオプションのダイアログ

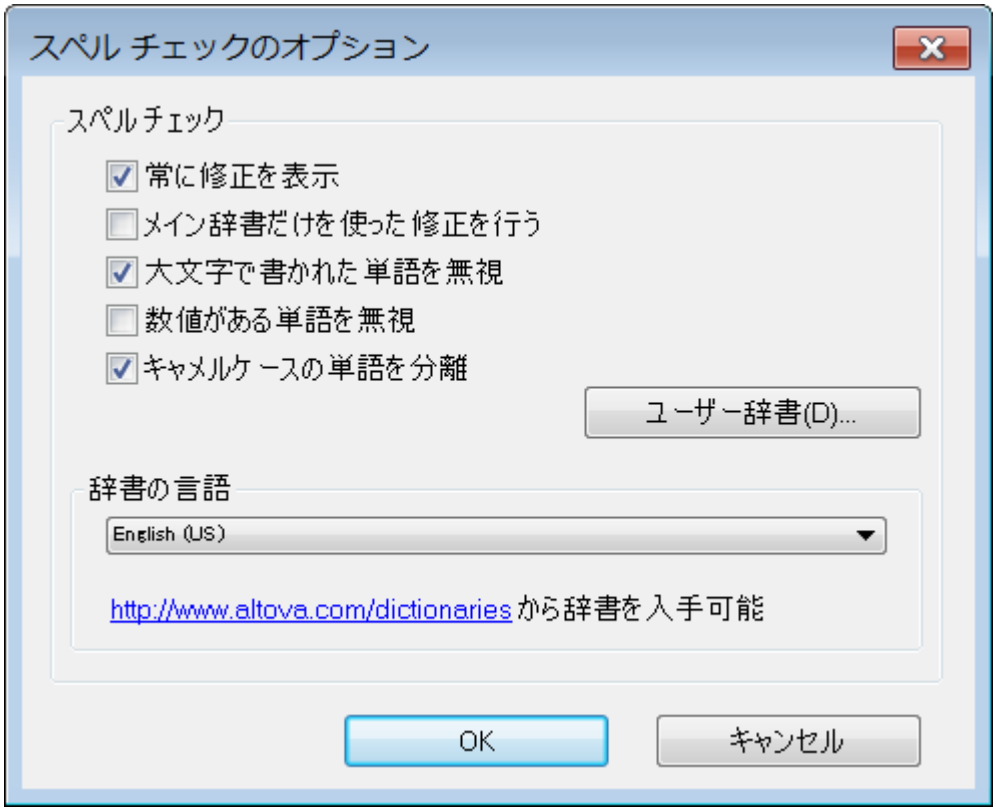
テキストビューまたはグリッドビューで「**スペルチェックのオプション**」コマンドをクリックすると、スペリングコンテキストダイアログが表示され、スペルチェックを行う範囲を指定することができます(以下のスクリーンショットを参照)。チェックを行う要素や属性を選択できるほか、CDATA セクションやコメントをチェックするか指定することができます。

スペルチェックを行う対象、またはスペルチェックを行う対象から除外する要素や属性を指定することができます。スペルチェックを全ての要素や属性に対して行うこともできます。

ダイアログ下部にある「**詳細スペリングオプション**」ボタンをクリックすることで、スペルチェックのオプションダイアログが表示されます。

スペルチェックのオプション

スペルチェックのオプションダイアログではグローバルなスペルチェッカーオプションを設定することができます。



常に修正を表示:
このオプションを選択することで、ユーザー辞書とそれ以外の辞書から得られた単語の候補が候補リスト ボックスに表示されます。このオプションを無効にすると、候補が表示されなくなります。

メイン辞書だけを使った修正を行う:
このオプションを選択することで、言語辞書（メイン辞書）だけが使用されます。提案にユーザー辞書は使用されません。「**ユーザー辞書**」ボタンも無効化され、ユーザー辞書の編集を行うことができなくなります。

大文字で書かれた単語を無視:
このオプションを選択することで、すべて大文字で記述された単語がチェックの対象から外されます。

数値を伴う単語を無視:
このオプションを選択することで、数値を含んだすべての単語が無視されます。

キャメルケースの単語を分離:
単語内に大文字の文字が含まれる単語をキャメルケースの単語といいます。例えば"CamelCase" という単語では、"Case" の"C" が大文字で示されているため、キャメルケースの単語として認識されます。キャメルケースの単語は通常辞書に含まれていないため、スペルチェッカーによりエラーとして認識されます。*キャメルケースの単語*オプションにより、キャメルケースの単語を大文字からはじまるパーツに分離して、個々のパーツをチェックすることによりこの問題を回避することができます。このオプションはデフォルトでチェックされています。

辞書の言語:
このコンボ ボックスではスペルチェッカーにて使用される辞書の言語が選択されます。デフォルトの選択はEnglish (US) となっています。Altova Web サイトからはその他の言語の辞書を無料でダウンロードいただけます。

(C) 2015-2021 Altova GmbH

スペルチェッカーに辞書を追加

辞書の言語は、.aff ファイルと.doc ファイルという2 つのHunspell 辞書ファイルから構成されます。言語辞書は、以下の場所にある Lexicons フォルダーへインストールされます。全ての言語の辞書は次の場所にある Lexicons フォルダー内にインストールされます: `C:\ProgramData\Altova\SharedBetweenVersions\SpellChecker\Lexicons`.

Lexicons フォルダー内部には、異なる言語に対応した辞書のファイルが、<言語名>¥辞書ファイル> という構成で各フォルダーに収められます。例えば、2つの(`English (British)` と`English (US)`) 英語辞書が以下のように配置されます:

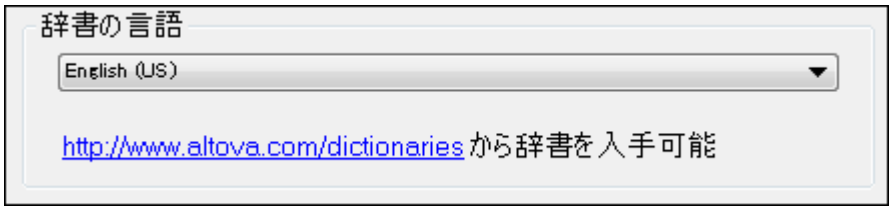
```
C:\ProgramData\Altova\SharedBetweenVersions\SpellChecker\Lexicons\English (British)
\en_GB.aff
C:\ProgramData\Altova\SharedBetweenVersions\SpellChecker\Lexicons\English (British)
\en_GB.dic
C:\ProgramData\Altova\SharedBetweenVersions\SpellChecker\Lexicons\English (US)\en_US.dic
C:\ProgramData\Altova\SharedBetweenVersions\SpellChecker\Lexicons\English (US)
\en_US.dic
```

スペルチェッカーオプション ダイアログでは、「辞書の言語」コンボボックスにあるドロップダウンリストに利用可能な辞書の言語が表示されます。Lexicons フォルダー以下にある言語サブフォルダーの名前がリストには表示されます。例えば上にある英語辞書の場合、English (British) とEnglish (US) という名前がリストに表示されます。

インストールされた辞書はコンピューターを使っているすべてのユーザー、そして(32 ビットか64 ビットかにかかわらず) 複数のAltova 製品により共有されます。

スペルチェッカーで使用される辞書は以下に示される2 種類の方法で追加することができ、どちらの方法でもファイルをシステムに登録する必要はありません

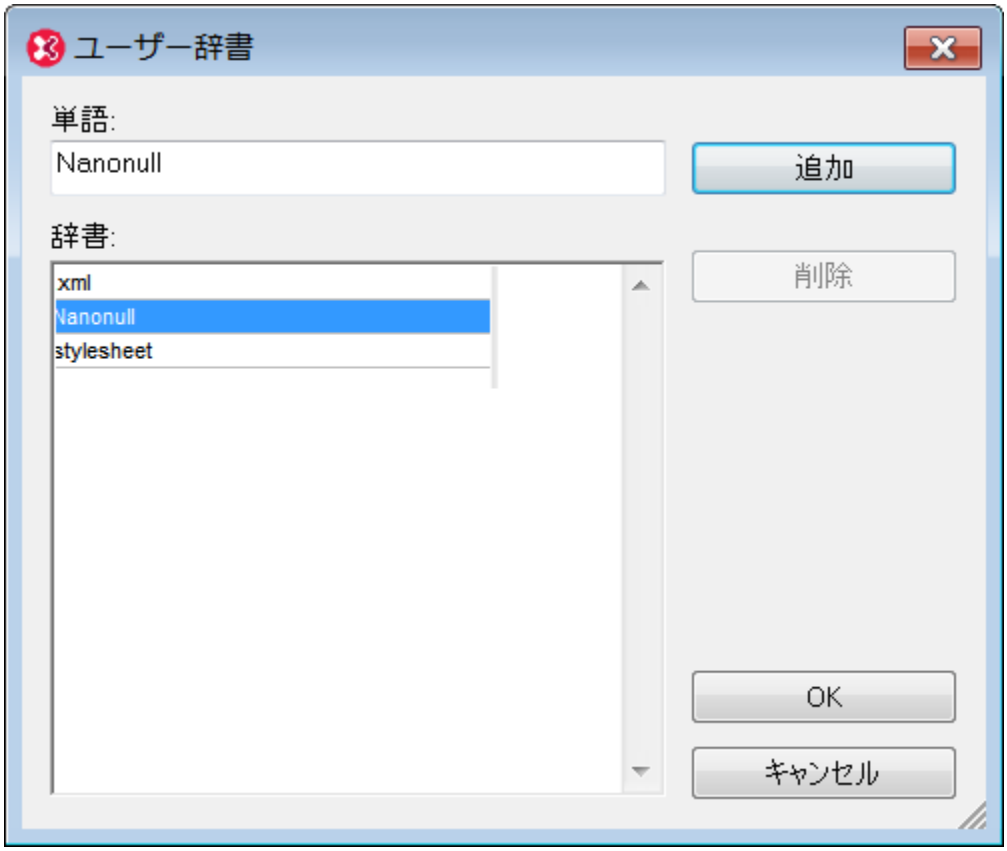
- Hunspell 辞書をLexicons フォルダーの新規サブフォルダーへ追加する。Hunspell 辞書は、例えば http://wiki.services.openoffice.org/wiki/Dictionaries や http://extensions.services.openoffice.org/en/dictionaries.といった場所からダウンロードすることができます (OpenOffice ではOXT という拡張子のZIP アーカイブが配布されているため、ファイル拡張子を.zip へ変換して、回答した.aff と.dic ファイルを ファイルをLexicons フォルダー以下にある言語フォルダーへコピーしてください。Hunspell 辞書はMyspell 辞書をベースにしているため、Myspell 辞書を使用することもできます)
- 複数の言語辞書をコンピューターの適切な場所へインストールするAltova 辞書インストーラーを使用する。スペルチェックのオプションダイアログにあるリンクをクリックすることで、インストーラーをダウンロードすることができます(以下のスクリーンショットを参照)。



メモ　辞書に対して適用されるライセンスの条項に同意するか、そしてお使いのコンピューターにおける辞書の使用が適切なものかの判断は、お客様へ委ねられます。

ユーザー辞書と作業

コンピューター上の各ユーザーには、ユーザーにより許可された単語を収めるためのユーザー辞書が与えられます。スペルチェックを行うと、言語辞書とユーザー辞書に含まれている単語のリストに対してドキュメント内の単語がチェックされます。ユーザー辞書ダイアログでは、ユーザー辞書に単語を追加、また削除することができます(*以下のスクリーンショットを参照*)。スペルチェックのオプションダイアログにて「ユーザー辞書」ボタンをクリックすることで、この辞書へアクセスすることができます(*このセクションの2番目のスクリーンショットを参照してください*)。



ユーザー辞書へ単語を追加するには、単語フィールドに単語を入力し、「**追加**」ボタンをクリックしてください。入力された単語がアルファベット順で辞書ペインに追加されます。辞書から単語を削除するには、目的の単語を辞書ペインにて選択し、「**削除**」ボタンをクリックしてください。その単語が辞書ペインから削除されます。ユーザー辞書ダイアログにおける編集を終えた後には「**OK**」をクリックすることで、変更点がユーザー辞書へ保存されます。

スペルチェックの際に単語をユーザー辞書へ追加することもできます。 スペルチェックダイアログ 974 は取るアクションのためのプロンプトをポップアップします。「**辞書に追加**」ボタンをクリックすると、未知の単語がユーザー辞書へ追加されます。

ユーザー辞書は以下の場所にあります: `C:\Users\<user>\Documents\Altova\SpellChecker\Lexicons\user.dic`

24.14.3 スクリプトエディター

スクリプトエディターコマンドにより、スクリプトエディターウィンドウが表示されます。スクリプトエディターの使用方法については、ドキュメンテーションの スクリプト環境 1056 セクションを参照ください。

メモ: スクリプトエディターを動作させるには、.NET Framework のバージョン 2.0 以降を、お使いのコンピューターにインストールする必要があります。

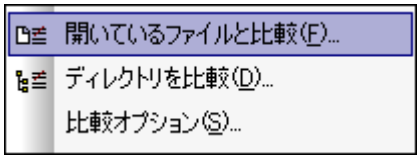
24.14.4　マクロ

マクロコマンドにマウスオーバーすることで、XMLSpy にて現在アクティブなスクリプトプロジェクトにて定義されているマクロがサブメニューに表示されます（以下のスクリーンショットを参照）。

サブメニューにあるマクロをクリックすることで、マクロが開始されます（上のスクリーンショットを参照）。

24.14.5　比較

XMLSpy には比較（または差分）機能が搭載されており、XML やテキストファイル、フォルダー間の比較を行うことができます。



ツールメニューには、以下のアイテムが用意されており、ファイルやフォルダーに対して比較処理を行うことができます:

- 開いているファイルと比較 983
- ディレクトリを比較 987
- 比較オプション 990

以下のサブセクションでは、これらのコマンドに関する記述を行います。

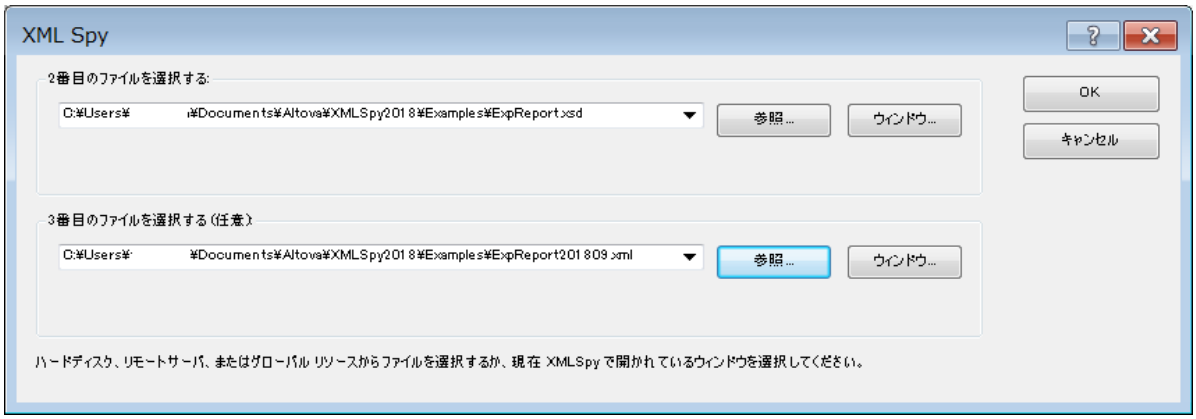
24.14.5.1　開いているファイルと比較

このコマンドによりファイルを他のファイルと比較することができます。比較は比較されるファイルをメインウィンドウ内で左右に表示し、ファイルの間に差分をハイライトして表示します。差分が2つのファイル間である場合、ハイライトされる差分は緑で表示されます。コンテンツが3つのファイル間で異なる場合（この機能は *Enterprise Edition のみで使用することができます*）差分はピンクで表示されます。（タグの構造とセマンティクスが顕著な場合）ファイルを XML ドキュメントとして、または、テキストドキュメントとして比較することができます。

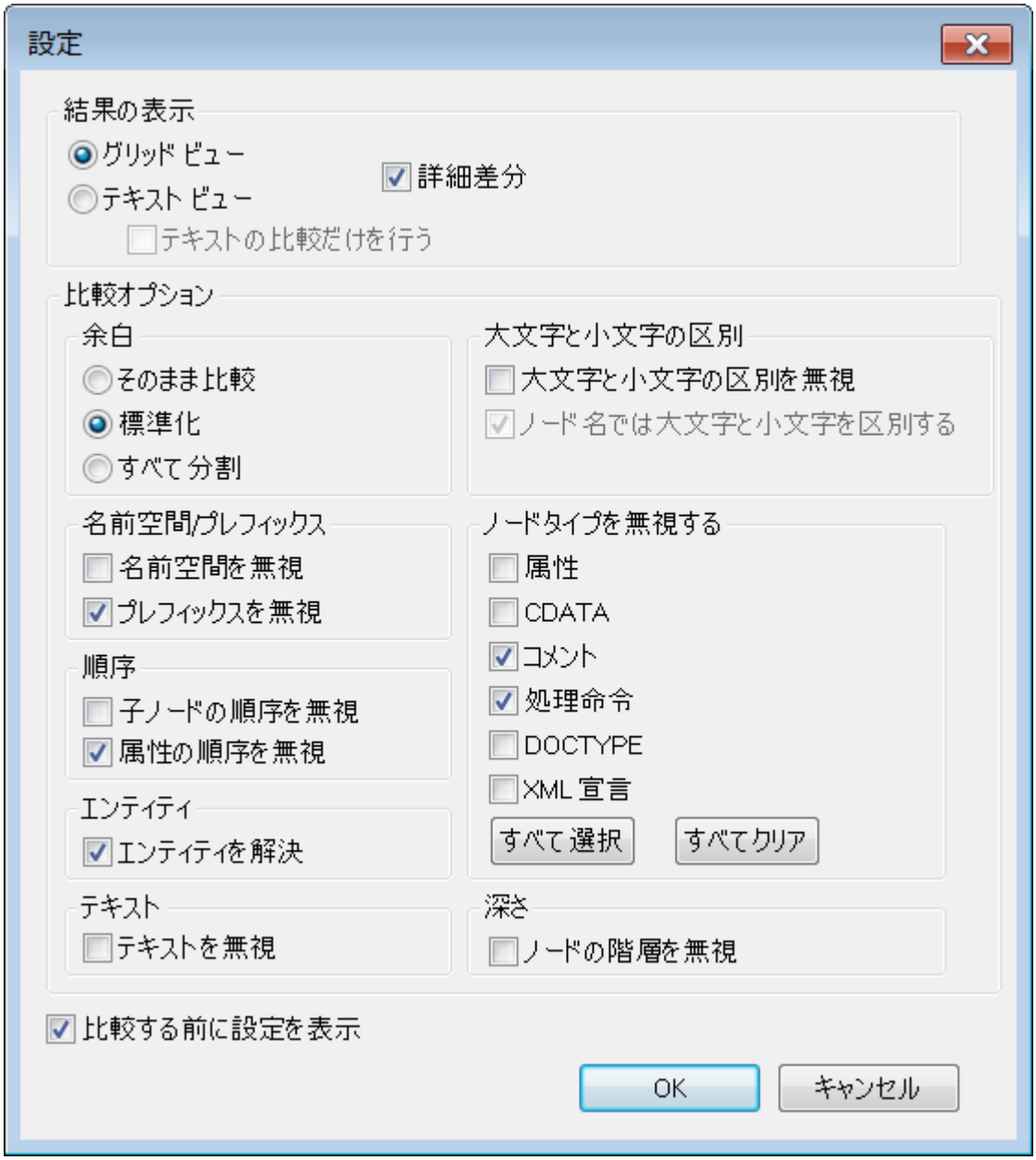
メモ　3 方向 ファイルの比較は Enterprise Edition のみで使用することができます。

（GUI 内で）アクティブなファイルを他のファイルと比較する場合、以下をおこないます:

1. メインウィンドウ内でアクティブなファイルを比較するには「**ツール | 開いているファイルと比較**」をクリックします（1度にアクティブに設定できるファイルは1つです）。アクティブなファイルと比較することのできる1つまたは2つのファイルを参照するダイアログが表示されます（*下のスクリーンショット*）。



2. ファイルエクスプローラー、グローバルリソース、またはURL を使用して 参照 746 をクリックしてファイルを参照します。「ウィンドウ」をクリックして、XMLSpy のウィンドウ内で開かれているファイルを選択します。
3. 「OK」をクリックします。設定ダイアログが表示されます（*下のスクリーンショット*）。これらの設定は、トピック比較オプション 990 内で説明されています。このダイアログを比較セッションが開始される都度開かないようにするには、（ダイアログの下にある）「比較の開始前に設定を表示する」チェックボックスのチェックを解除してください。



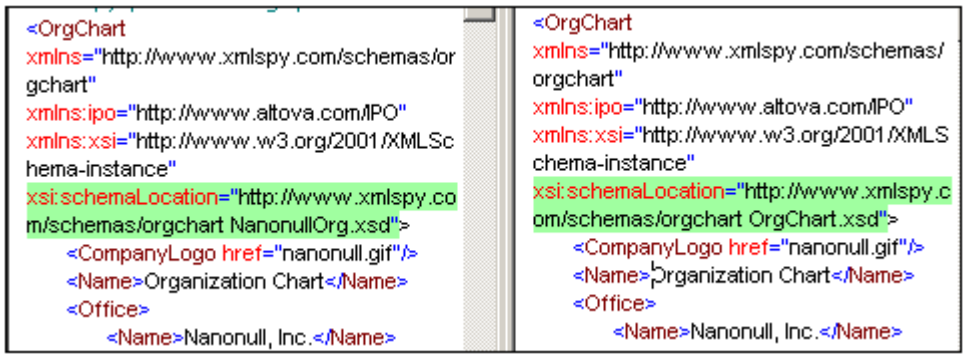
4. 必要な設定を選択し、「OK」をクリックします。次の2つの状態が発生します: (i) 個別のペイン内に比較するファイルが左右に並べられて表示されます。(ii) ファイルの比較 管理ウィンドウが表示されます。(次のセクション *ファイルの比較* 986 *のスクリーンショット参照*)。
5. ファイルの比較 管理ウィンドウ内のボタンを使用して差分をナビゲートしマージします 。(*使用方法に関しては ファイルの比較* 986 *を参照してください。*)
6. 差分のレビューまたはマージが終わると、「完了」をクリックしてください。

メモ　(「ツール | 比較オプション」990 選択することにより)比較セッション中に比較設定が変更される場合があります、影響は次の比較セッションからです。新規の設定は現在のセッションに影響を与えません。

ファイルの比較と差分のマージ

2つの種類の比較が存在します:

- *2 方向比較:* 2つのファイルが比較されます。左側のペイン内にアクティブなファイルが表示され、2番目のファイルに対して比較されます(*下のスクリーンショットを参照*)。



- *3 方向比較 (Enterprise Edition のみ):* 3つのファイルが比較されます。左側のペイン内にアクティブなファイルが表示されます。比較のために選択された2つの最初のファイルが中央のペインに表示されます。この中央のファイルがベースファイルと称されます。左側のファイルと中央のファイルの差分をマージする、または、右側のファイルと中央のファイルの差分をマージすることもできます。

メモ ファイルの比較管理ウィンドウ内のボタンが正確に対応しない場合があるためウィンドウ内でペインを移動しないでください。

差分と競合

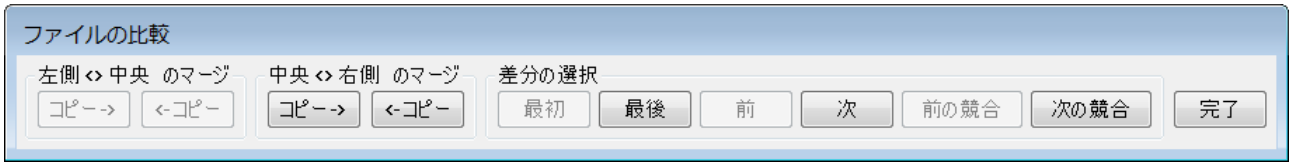
差分と競合は異なる色でハイライトされ、個別にナビゲートすることができます(*下の ハイライト色* 987 *を参照*)。

- *差分:* 対応する場所の2つのファイル内のコンテンツが異なる場合。緑色でハイライトされます。
- *競合:* 対応する場所の3つのファイル内のコンテンツが異なる場合。ピンク色でハイライトされます。

メモ 2 方向比較は競合は表示せず、差分のみを表示することに注意してください。

差分と競合のナビゲート

ファイルの比較 管理ウィンドウの差分の選択ペイン内のボタンを使用してドキュメント内を移動することができます(*下には 2 方向比較と 3 方向比較が表示されています*)。ペインには2 方向比較のための4つのボタンと、3 方向比較のための6つのボタンが含まれています(*スクリーンショットを参照*)。



ファイルの比較 管理ウィンドウには次のボタンが含まれます:

- *最初:* ドキュメント内の全ての差分と競合の最初に移動します。
- *最後:* ドキュメント内の全ての差分と競合の最後に移動します。

- *前:* 現在アクティブな差分または競合から前の差分に移動します。
- *次:* 現在アクティブな差分または競合から次の差分に移動します。
- *前の競合:* 現在アクティブな差分または競合から前の競合に移動します。
- *次の競合:* 現在アクティブな差分または競合から次の競合に移動します。

差分と競合のマージ
マージペイン内のボタンをしようしてハイライトされたコンテンツを他のペインにコピーします。適切なペインの組合せのコピーの方向を使用してください。マージを有効化するには、次の比較オプション 990 が満たされている必要があります:

- 詳細差分がチェックされており、
- ノードの階層を無視のチェックが解除されていなければなりません

差分のレビューとマージが終わると、「**完了**」をクリックします。

メモ　マージする場合、比較セッションは停止し、変更を元に戻すファイルを選択して「**編集 | 元に戻す**」を選択し、「Ctrl+Z」を押してください。

メモ　比較セッションがアクティブな間、ビューの編集または変更は許可されていません。ビューの変更を試みると、比較セッションの中断を警告するメッセージが表示されます。

ハイライト色

2つのファイル間で対応するラインが異なる場合、両方のファイル内で差分は緑でハイライトされます。3つのファイル間でラインが異なる場合は全てのファイル内で差分がピンク色でハイライトされます。差分が選択されている場合、両方のファイルで濃い緑色で表示されます。競合が選択される場合、濃いピンク色で表示されます。

比較セッション内で使用されるハイライト色は以下のとおりです。

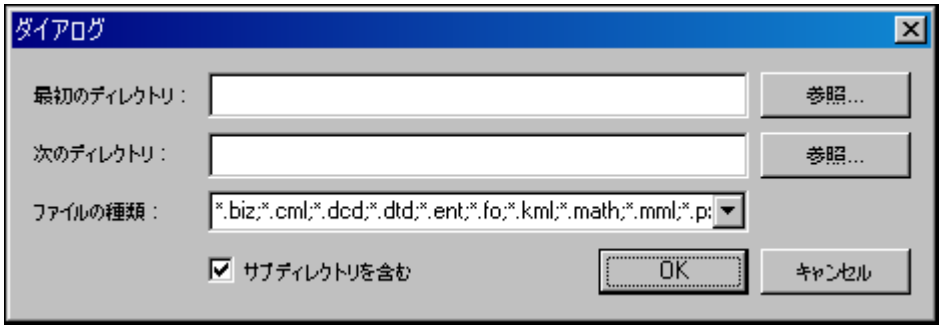
	差分
	現在の差分
	マージされた差分
	現在のマージされた差分
	競合
	現在の競合

24.14.5.2 ディレクトリの比較

ディレクトリの比較コマンドにより、2つのディレクトリをサブディレクトリを含めて、または含めないで比較することができます。ディレクトリの比較では、一方には無いファイルや、同名のファイルでも内容が同一であるかどうかが表示されます。

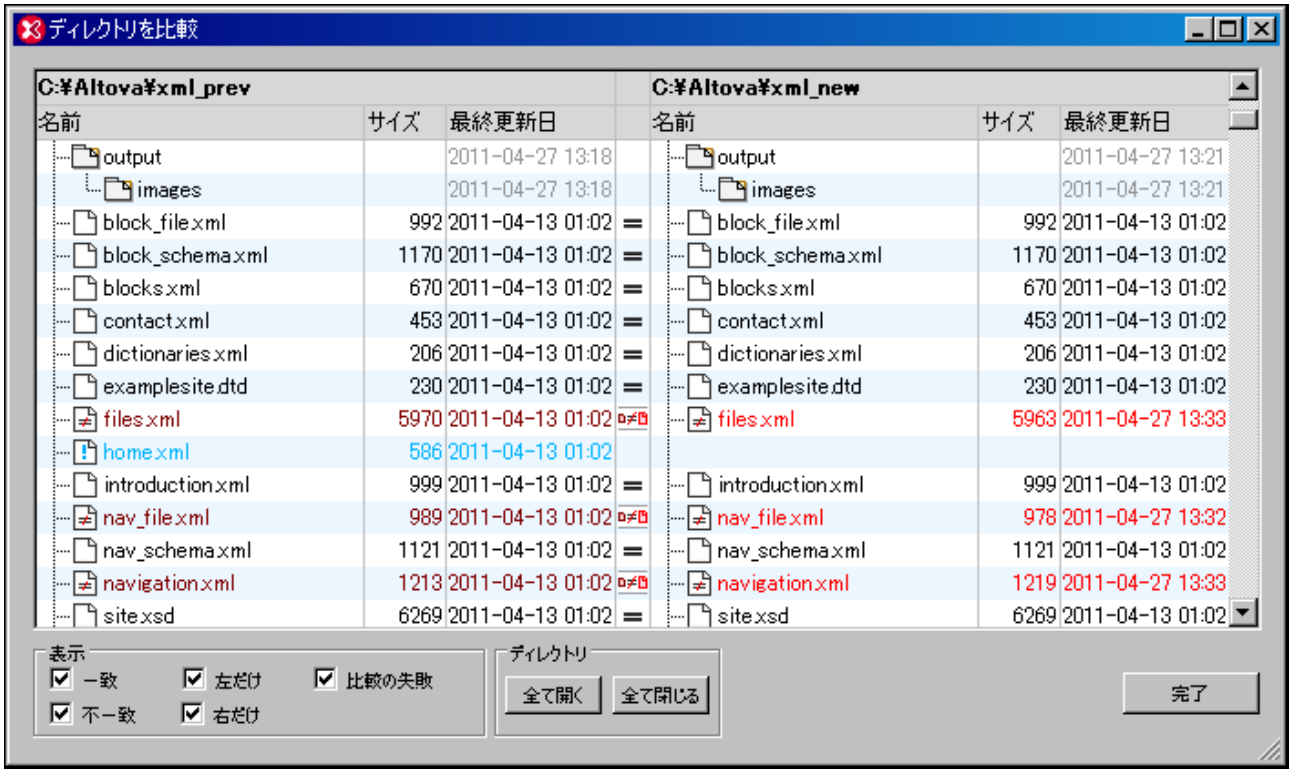
2つのディレクトリを比較するには:

1. 「**ツール | ディレクトリの比較**」をクリックします。以下のダイアログが表示されます:



2. 比較するディレクトリを指定します。比較する対象にサブディレクトリも含める場合は、**サブディレクトリを含む**チェックボックスにチェックを入れてください。
3. **ファイルの種類**フィールドにて、比較を行いたいファイルの種類を選択します。ドロップダウンメニューからは、(i) XML ファイルタイプ、(ii)(ii) XMLSpy にて定義されたファイルタイプ[1018] (iii) 全てのファイルタイプを選択することができます。
4. 「OK」をクリックします。(比較オプション[990] にて記述される) 設定ダイアログが表示されます。
5. ファイルの比較を行うための設定を選択します。
6. 「OK」をクリックします。比較の進捗状態を示すダイアログが表示されます。

比較の結果が、以下に示されるウィンドウに表示されます:



ディレクトリシンボル

ディレクトリの名前は、全て黒色で表示されます。

ディレクトリは縮退されており、コンテンツは表示されていません。

ディレクトリは展開されており、コンテンツがその行以下に表示されます。

ディレクトリにはファイルが含まれており、中に含まれるファイルは、比較することができないか、比較対象となるファイルと同一のものです。

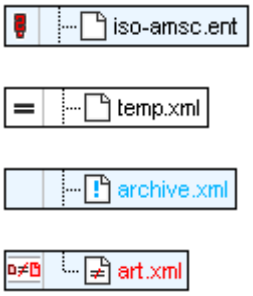
比較対象のディレクトリには含まれていないファイルが、このディレクトリに含まれています。

比較対象のディレクトリに含まれているファイルとは内容が異なるファイルがこのディレクトリに含まれています。

比較対象のディレクトリには含まれていないファイルが1つ以上、また、比較対象のディレクトリに含まれているファイルとは内容の異なるファイルが1つ以上、このディレクトリには含まれています。

ファイルシンボル

比較ステータスにより、ファイル名の表示に使用される色が以下のように変化します:



比較対象となるファイルと比較を行うことができません。中間のカラムに疑問符が表示されます。ファイル名は黒色で表示されます。

比較対象となるファイルと同一のファイルです。中間のカラムに等号記号が表示されます。ファイル名は黒色で表示されます。

比較対象となっているディレクトリに、このファイルは存在しません。中間のカラムには何も表示されず、ファイル名は水色で表示されます。

比較対象となるファイルと内容が一致せず、比較対象のファイルと比べて、最終更新日時が新しいことを表します。(最終更新日時が新しい方のファイルは明るい赤色で表示され、中間カラムにあるアイコンも、新しいファイルがある方向に明るい赤色が表示されます。

表示オプション

- 結果ウィンドウの下部にある表示ペインにあるチェックボックスを操作することで、表示するファイルの種類の選択することができます.
- ディレクトリペインにあるボタンをクリックすることで、全てのサブディレクトリを展開、または縮退することができます。
- フォルダーアイコンをダブルクリックすることで、サブディレクトリを展開または縮退することができます。
- ファイルのタイトルバーを右クリックして、サイズと最終更新日をクリックすることで、サイズと最終更新日の表示を切り替えることができます。

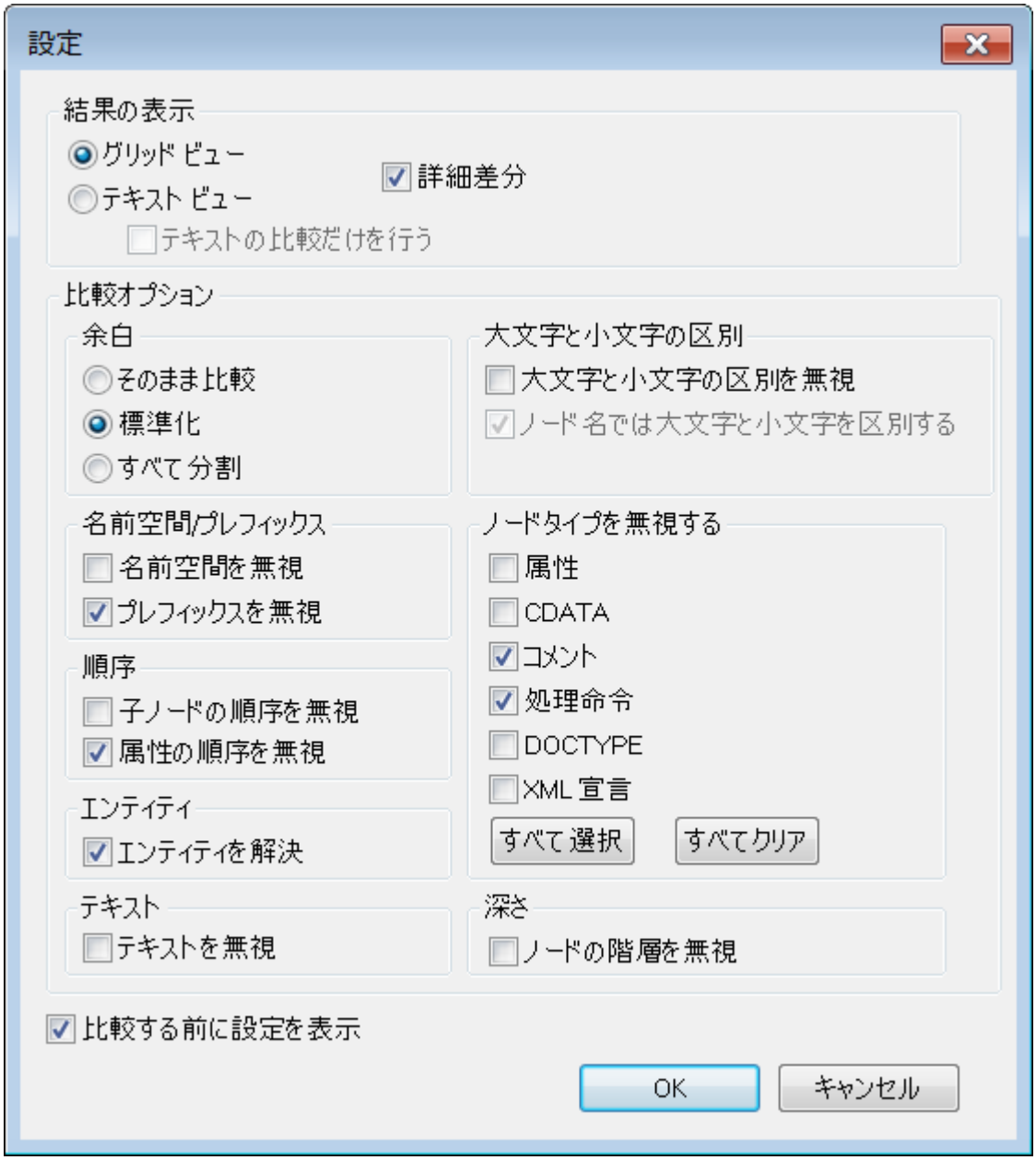
- カラムをドラッグすることで、カラムの幅を変更することができます。
- 結果ウィンドウは、移動、最小化、そして大きさを変更することができます。

ファイルの比較とマージ

結果ウィンドウに表示されている行をダブルクリックすることで、両方のファイルがメインウィンドウで開かれ、両方のファイルに対してファイルの比較が行われます。その後は、通常の比較セッションと同様に進めることができます（ 開いているファイルと比較… 983)を参照ください）。

24.14.5.3 比較オプション

「**ツール | 比較オプション**」を選択することで、設定ダイアログを開くことができます（以下のスクリーンショットを参照）。設定ダイアログでは、比較セッションの設定を指定することができます。設定ダイアログで指定されたオプションが、その比較セッションで使用されます。



結果の表示

比較の結果が表示されるビューを、以下のオプションから選択します:

- グリッドビュー(XML 比較)
- テキストの比較だけを行う、が無効なテキストビュー(XML 比較)
- テキストの比較だけを行う、が有効なテキストビュー(テキスト比較)

XML の比較結果を表示することができるビューを選択すると、ドキュメントはXML ドキュメントとして扱われ、XML 比較オプションが有効になります。テキスト比較が選択された場合は、テキストの比較でも使用することのできるオプション(余白と大文字/小文字の比較)だけが有効になり、その他のオプションは無効になります。

メモ: 差分のマージは、テキストビューならびにグリッドビューで、そしてXML とテキストの比較モードで行うことができます。マージの取り消しを行いたい場合は、比較セッションを中止して、変更されたファイルを選択し、「**編集 | 元に戻す**」を選択するか、**Ctrl + Z** を押下してください。

詳細差分

このオプションが選択されていない場合、すぐそばにある兄弟要素も1つの差分として認識され、マージオプションが無効になります。このオプションが選択されている場合、すぐそばにある兄弟要素は異なる差分セグメントとして認識され、マージオプションが有効になります。

メモ: マージを行うには、「**詳細差分**」チェックボックスにチェックを入れる必要があります。

余白

余白として認識される対象は、タブならびに改行です。このオプションにより、ファイルを変更することなく余白の正規化を行うことができ、連続した余白が1つの余白としてみなされ、全ての余白が取り除かれます。余白オプションはXML とテキスト比較の両方に対して使用することができます。

大文字と小文字の区別

「**大文字と小文字の区別を無視**」チェックボックスにチェックが入れられていない場合、(XML の比較において)ノード名で使われている文字列の大文字と小文字を区別するか指定することができます。大文字と小文字の区別オプションは、XML とテキスト比較の両方に対して使用することができます。

名前空間/プレフィックス

これらのオプションにより、差分の検索を行うにあたり、名前空間やプレフィックスを無視するか指定することができます。

順序

子ノードの順序を無視オプションが有効になっている場合、子ノードの順序は比較の際に考慮されなくなります。比較は子ノード全体に対して行われ、ドキュメント間にある子ノード同士の違いが、相対的な出現箇所だけである場合、それらの違いは無視されます。

メモ: 各子要素ノードは、その名前、属性、そして位置により特定されます。あるノードの名前が、兄弟要素の名前と同じ場合、これらのノードを特定するために(「子ノードの順序を無視」オプションが有効になっている場合でも)ノードの位置情報が使用されます。このオプションは、ドキュメントの各レベルに対して個別に適用されます。

子ノードの順序を無視オプションが無効になっている場合、ノードの順序による違いも差分として認識されます。属性の順序を無視オプションも利用することができ、各要素にある属性の順序に対して適用されます。

エンティティ

エンティティを解決オプションが有効になっている場合、ドキュメントにある全てのエンティティが解決されます。そうでない場合、ドキュメントにあるエンティティ名が、そのままの状態で比較されます。

テキスト

「**テキストを無視**」オプションが選択されている場合、対応するテキストノード間の比較は行われません。

無視するノードタイプ

比較セッションにて比較の対象には含まれないノードの種類にチェックを入れてください。無視することのできるノード型は、属性、CDATA、コメント、処理命令、DOCTYPE ステートメント、そしてXML 宣言です。

(C) 2015-2021 Altova GmbH

深さ

「ノードの階層を無視」にチェックが入っていると、任意の要素における子ノード（子孫ノード）が無視されます。マージを行うには、このオプションを無効にする必要があります。

比較する前に設定を表示

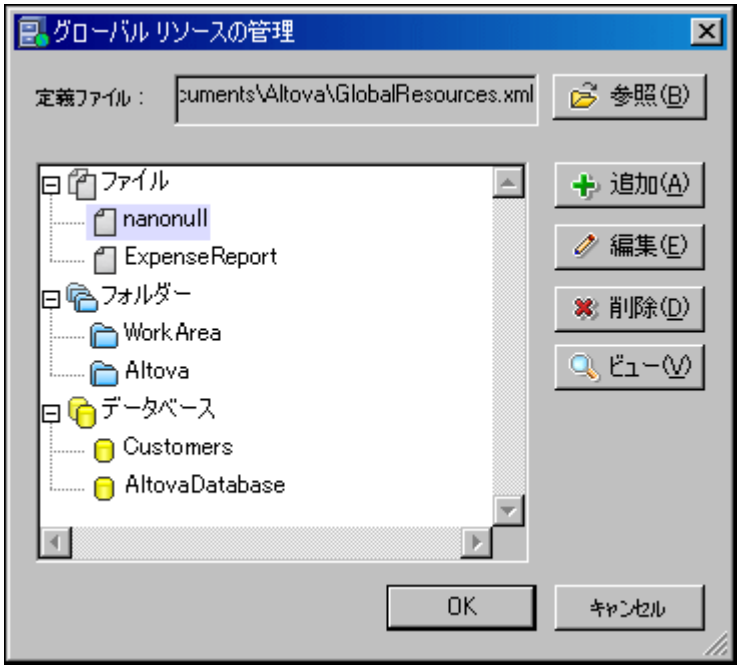
このオプションにチェックを入れることで、「開いているファイルと比較」コマンドが選択されるたびに、設定ダイアログが表示されます。比較を行う前に設定ダイアログを表示することで、比較設定のチェックや変更を、各比較に対して行うことができます。

このコマンドがチェックされていない場合、比較コマンドが指定された直後に、比較セッションが開始されます。

24.14.6　グローバルリソース

グローバルリソースコマンドにより、以下の操作を行うことができるグローバルリソースダイアログが表示されます（以下のスクリーンショットを参照）：

- グローバルリソースXML ファイルを指定。
- ファイル、フォルダー、データベースのグローバルリソース（またはエイリアス）を追加。
- 各グローバルリソース（エイリアス）に対して、各種構成を指定。各構成により、特定のリソースへのマッピングが行われます。

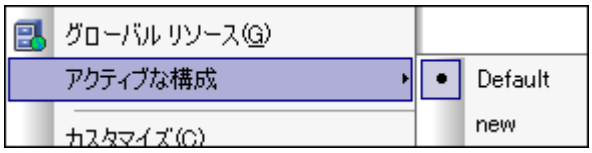


グローバルリソースの定義方法に関する詳細は、グローバルリソースの定義 660 セクションを参照ください。

メモ　Altova グローバルリソースダイアログへは、グローバルリソースツールバー 1000（「**ツール | カスタマイズ | ツールバー | グローバルリソース**」）からアクセスすることができます。

24.14.7 アクティブな構成

アクティブな構成メニューアイテムにマウスオーバーすることで、現在アクティブなグローバルリソースXML ファイル[993]にて定義されている全ての構成が表示されます(以下のスクリーンショットを参照)。

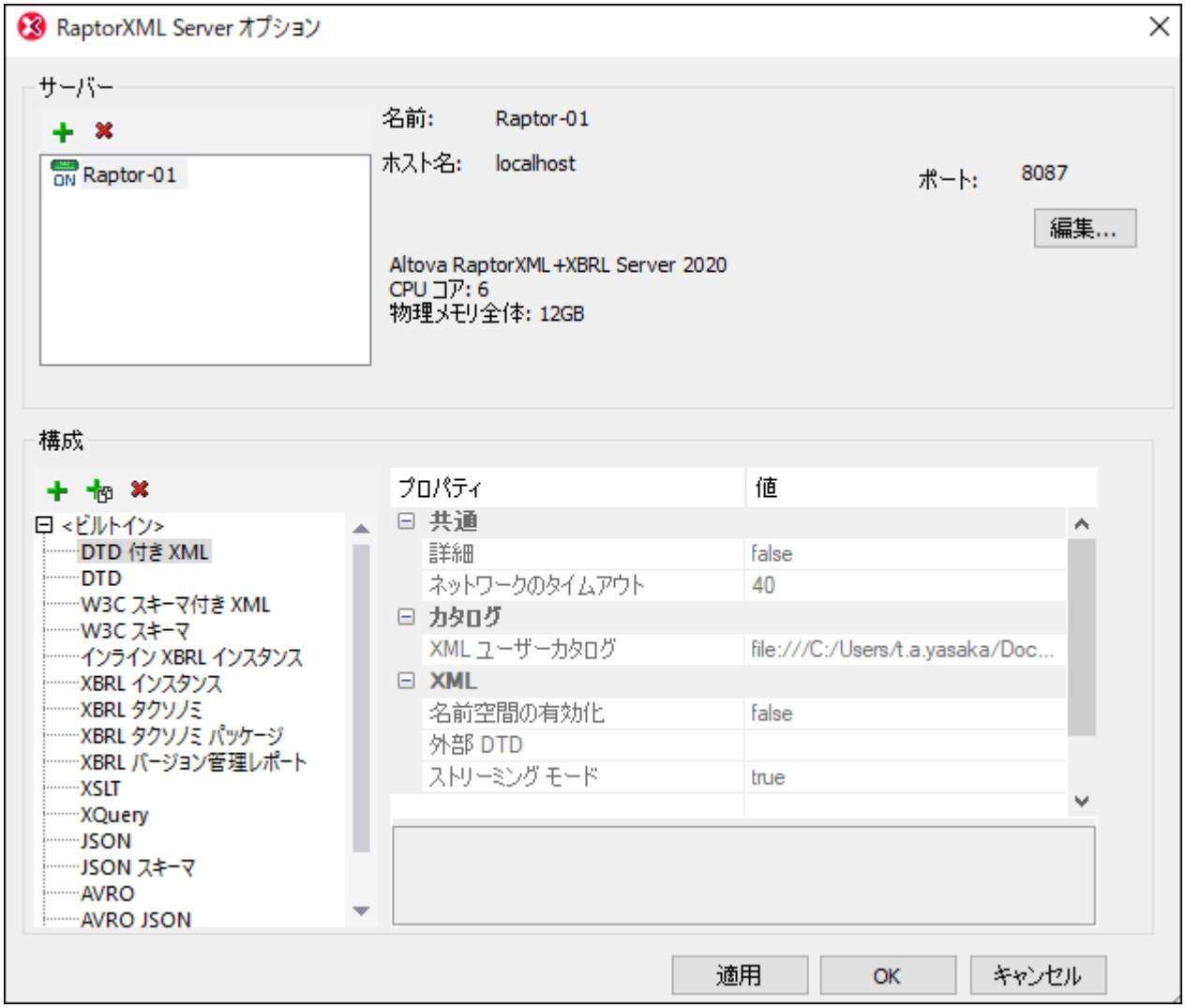


現在アクティブな構成は、中黒(・)により表示されます。上のスクリーンショットでは、現在アクティブな構成が Default となっています。アクティブな構成を変更するには、アクティブにしたい構成を選択してください。

メモ: アクティブな構成は、グローバルリソースツールバー[1000](「**ツール | カスタマイズ | ツールバー | グローバルリソース**」)から選択することができます。

24.14.8 Raptor Server の管理

Raptor Server の管理 コマンドにより、Raptor Servers 使用することのできる Raptor Servers のプールに追加し、各サーバーのために複数の構成を定義することができます。XML と XBRL ドキュメントを検証するために RaptorXML Server を使用する場合は、RaptorXML Server を使用して検証[683]を参照してください。



Raptor Server の追加

ダイアログの *サーバーペイン*（*下のスクリーンショット*）内から、「**サーバーの追加**」アイコンをクリックします。アイコンをクリックして、Raptor Server、がインストールされているマシンの名前(ホスト名)、およびRaptor Server のポートを認識するための名前を入力します。「OK」をクリックして設定を保存します。

- *名前:* 選択されるすべての文字列。XMLSpy 内では、特定のRaptorXML Server を認識するために使用されています。
- *ホスト名:* Raptor Server がインストールされているマシンのネットワークの名前 またはIP アドレス。処理はホスト名よりも IP アドレスを使用した場合のほうが速くなります。`localhost`（ローカルマシン）に対応するIP アドレスは`127.0.0.1` です。
- *ポート:* Raptor Server がアクセスされるポート。このポートは、（`server_config.xml` という名前の）Raptor の構成ファイル内で指定されています。ポートは、固定され、サービスへのリクエストが対応できるように知らされている必要があります。Raptor 構成ファイルの詳細は、*RaptorXML Server* および *RaptorXML+XBRL Server* のユーザーマニュアルを参照してください。

サーバー情報を入力すると、「OK」をクリックします。入力したサーバー名が（ペインの左側の）サーバーリストに表示されます。サーバーの名前の横に緑のアイコンが表示され、サーバーが開始され、実行中であることを示しています。サーバーの詳細はペインに表示されています（*上のスクリーンショット参照*）。赤いアイコンはサーバーがオフラインであることを示します。サーバーを検出することができない場合、エラーメッセージが表示されます。

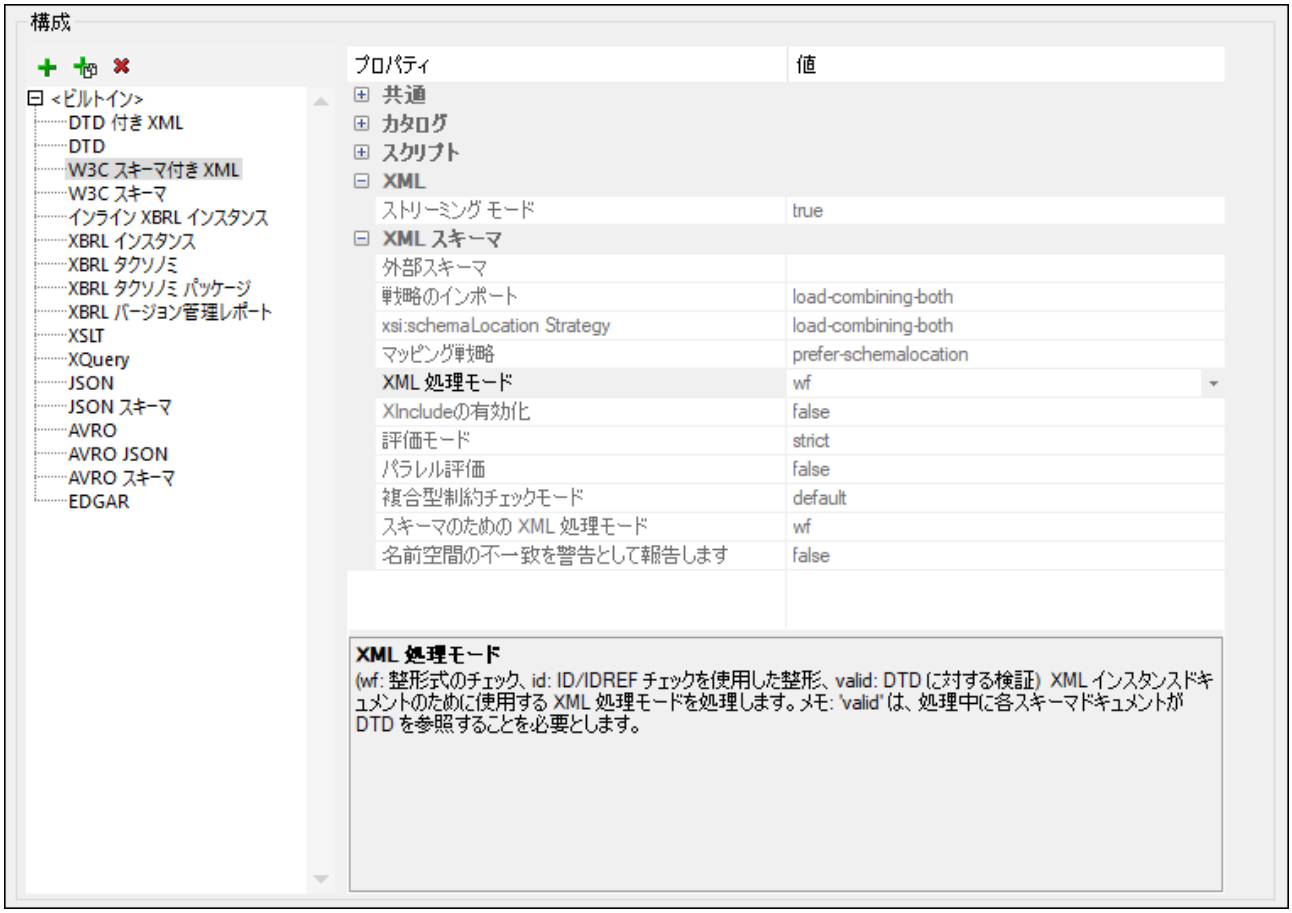
メモ Raptor Server は、サーバーが追加される際に実行中である必要があります。これはXMLSpy がサーバーの情報を取得し、保管するために必要です。サーバーが追加された後、サーバーがオフラインまたは検出されない場合、これらのシナリオは赤いアイコンまたはエラーメッセージで通知されます。

サーバーの名前、ホスト名、またはポートを編集する場合、左側のペインのサーバーを選択して、「**編集**」ボタンをクリックし、表示されるダイアログ内の情報を変更します。プールからサーバーを削除する場合は、サーバーを選択して「**選択されたサーバーを削除**」アイコンをクリックします。

Server 構成

構成とは、RaptorXML 検証オプションのセットです。サーバーが追加されると、構成を持ち、`default` と名前が付けられます。これは RaptorXML オプションのセットに設定されたデフォルトの値です。他のオプションの値を含む新しい構成を追加することもできます。複数のサーバー構成を追加した後、アクティブな構成になる1 つの構成を選択します。これが「**サーバー上で検証**」コマンドが実行される時に使用される構成です。

構成ペインは2 つの部分により構成されています: (i) それぞれが変更可能なドキュメント型のリストを含む構成を表示する左側のペイン、(ii) 左側のペインで選択されたドキュメント型のための検証オプションを表示する右側のペイン。右側のペインの下にはお選択されたオプションの説明が表示されています（*上のスクリーンショット参照*）。



構成の追加

RaptorXML Server オプションダイアログの構成ペイン内から(*上のスクリーンショット*)「**構成の追加**」をクリックします。デフォルトのオプション値に新規構成が追加されます。「**選択された構成をコピー**」をクリックすることにより新規構成を作成することもできます。これによりコピーされた構成のオプション値と同じ値を持つ新しい構成が作成されます。新たしい構成は、型 `config<X>` のデフォルトの名前と共に作成されます。構成の名前をダブルクリックして新しい名前を入力して編集します。構成のオプション値のすべてを編集することができます。

構成のオプション値の編集

左側のペインからドキュメント型を選択します。これにより右側のペイン内に選択されたドキュメント型の検証オプションが表示されます。オプションの値を編集するには、(オプションの値の型に従い)以下を行います:

- 値が定義済みの値のセットの1つの場合、そのオプションの値の列のコンボボックスから値を選択します。
- 値に制約がある場合、オプションの値フィールドをクリックして、値を入力します。
- 値がファイルパスの場合、値を入力できるだけでなく、オプションの値の列の「**参照**」ボタンを押して、使用するファイルを参照します。

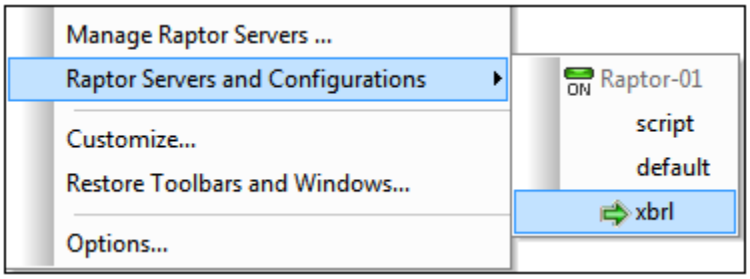
オプションを選択すると、右側のペインの下のボックスに詳細が表示されます。各オプションの詳細は、RaptorXML Server および RaptorXML(+XBRL) Server ユーザーマニュアルのコマンドラインインターフェイスのセクションを参照してください。

構成の削除

左側のペインから、削除する構成を選択して、「**選択された構成を削除**」をクリックします。

24.14.9 Raptor Server と構成

複数のサーバーで複数の構成を定義した場合、サーバーとそのアクティブな構成として1つの構成を選択することができます。アクティブな構成は、以降の検証に使用されます。カーソルを「ツール | Raptor Servers および構成」コマンドにポイントして、追加されたすべてのサーバーとその詳細が含まれるサブメニューから、を含む表示されるサブメニューが表示されます（*下のスクリーンショット参照*）。アクティブな構成と設定するサーバー構成を選択します。したのスクリーンショットでは、`Raptor-01` という名前のサーバーの(緑色の矢印で示されている) `xbrl` 構成がアクティブな構成として選択されています。



24.14.10 カスタマイズ

カスタマイズ コマンドにより、XMLSpy に搭載されたメニューやツールバーをカスタマイズすることができます。以下のタブが用意されています:

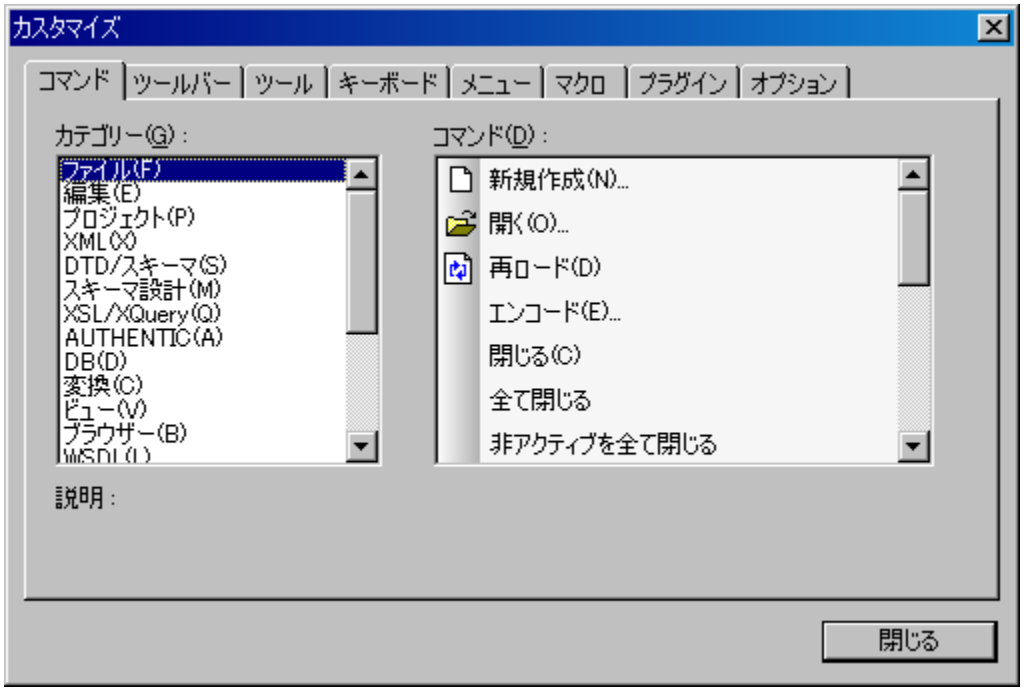
- コマンド[998]: 全てのアプリケーションやマクロコマンドをメニューバー、メニュー、ツールバーへドラッグします。
- ツールバー[1000]: 個々のツールバーを有効化、無効化、リセットすることができます。
- ツール[974]: 外部プログラムを開くコマンドをインターフェースへ追加することができます。
- キーボード[1003]: 個々のアプリケーションならびにマクロに対してキーボードのショートカットを作成することができます。
- メニュー[1006]: カスタマイズするメニューバーならびにコンテキストメニューを選択し、有効にすることができます。コマンドタブと一緒に操作します。
- マクロ[1008]: コマンドに対してマクロを関連付けることができます。
- プラグイン[1010]: プラグインの有効化と統合を行うことができます。
- オプション[1014]: ツールバーの表示オプションをセットすることができます。

このセクションではカスタマイズダイアログが開かれ、メニューバー、メニュー、またはツールバーアイテムが右クリックされた時に表示される コンテキストメニュー[1011] についても記述されます。

24.14.10.1 コマンド

コマンド タブでは、メニューやツールバーをカスタマイズすることができ、必要に応じてアプリケーションコマンドをメニューやツールバーに追加することができます。新たなアプリケーションコマンドやメニューを独自に作成することはできない点に注意してください。

(C) 2015-2021 Altova GmbH



以下の操作によりコマンドをツールバーやメニューへ追加することができます:

1. メニューオプションから「**ツール | カスタマイズ**」を選択します。カスタマイズダイアログが表示されます。
2. カテゴリーリストボックスから**全てのコマンド**カテゴリーを選択します。利用可能なコマンドがコマンドリストボックスに表示されます。
3. コマンドリストボックスに表示されたコマンドをクリックして、表示されているメニューやツールバーへドラッグします。許可されている場所へカーソルを移動させると、マウスポインターにI というアイコンが表示されます。
4. コマンドを挿入する場所でマウスボタンを離してください。

以下の点に留意してください:

- コマンドをドラッグすると、マウスポインターの端に小さなボタンが表示されます。これにより、ボタンがドラッグされていることが示されます。
- ポインター下部に表示された "x" は、現在マウスポインターがある位置にドロップできないことを表しています。
- コマンドをドロップすることができる位置(ツールバーまたはメニュー)にカーソルを移動すると、"x" が消えてI がというアイコンが表示されます。
- コマンドはメニューまたはツールバーに配置することができます。独自のツールバーを作成した場合[1000]した場合、このカスタマイズ機能を使用することでコマンドを配置することができます。
- カーソルをメニュー近くに移動させると、そのメニューが開かれ、メニュー内部にコマンドを配置することができるようになります。

コンテキストメニューへコマンドを追加

以下の操作により、コンテキストメニューにコマンドを追加することもできます:

1. カスタマイズダイアログにて メニュータブ[1006]をクリックします。
2. コンテキストメニューペインにてコンボボックスから目的のコンテキストメニューを選択します。選択されたコンテキストメニューが表示されます。
3. カスタマイズダイアログにてコマンドタブへ切り替えます。
4. コマンドリストボックスから追加するコマンドを選択し、コンテキストメニューまでドラッグします。

コマンドまたはメニューの削除

以下の操作により、メニューやコンテキストメニュー、またはツールバーからコマンドを削除する、またはメニュー全体を削除することができます:

1. メニューオプションの「**ツール | カスタマイズ**」を選択してカスタマイズダイアログを開きます。
2. カスタマイズダイアログが表示されている状態で、メニューまたはメニューコマンドを右クリックして、表示されるコンテキストメニューから**削除**を選択します。また、メニューやメニューコマンドを、"x" がマウスポインターに表示される場所までドラッグアンドドロップすることでもメニューまたはメニューコマンドを削除することができます。

このセクションに記述された方法により削除されたメニューコマンドは、復元することができます。削除されたメニューを復元するには、メニューオプションから「**ツール | カスタマイズ | メニュー**」を選択し、アプリケーションフレームメニューペインにある**リセット**ボタンをクリックしてください。その他にも、「**ツール | カスタマイズ | ツールバー**」を選択し、左側に表示されたリストからツールバーを選択し、**リセット**ボタンをクリックすることもできます。

24.14.10.2 ツールバー

ツールバータブでは、(i) 特定のツールバーを有効/無効にする(つまりどのツールバーがインターフェースにて表示されるかを指定する)、(ii) 各ツールバーに表示されるアイコンをセットする、そして(iii) 独自のツールバーを作成することができます。

ツールバーには頻繁に使用されるメニューコマンドが収められています。カーソルをアイコン上部に移動することで、そのアイコンに関する情報がツールチップとステータスバー上に表示されます。ツールバーは、フロート状態のウィンドウとして画面上の任意の場所へ配置することができます。

メモ: ツールバーへコマンドを追加するには、目的のコマンドを コマンド 998 タブにあるコマンドリストボックスからツールバーへドラッグする必要があります。ツールバーからコマンドを削除するには、カスタマイズダイアログが開かれた状態で、ツールバーからコマンドをドラッグしてください(詳細については コマンド 998 のセクションを参照ください)。

メモ: 特定のビューに対して定義されたツールバーの設定は、デフォルトでそのビューに対してだけ有効になっています。その設定を全てのビューに対して適用するには、ダイアログ下部にある**全てのビューに変更を適用**チェックボックスにチェックを入れてください。

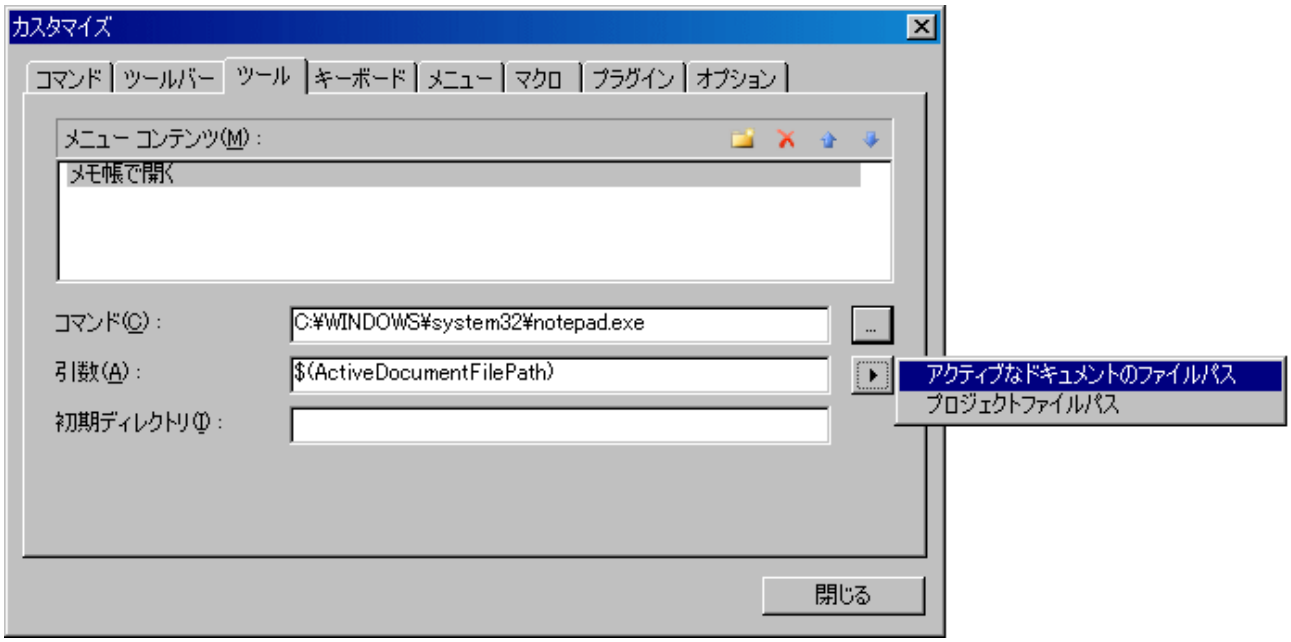
(C) 2015-2021 Altova GmbH

以下の機能を利用することができます:

- ツールバーの有効化/無効化: ツールバーリストボックス内に表示されたチェックボックスをクリックしてください。
- 全てのビューに変更を適用: ダイアログ下部に表示されているチェックボックスにチェックを入れてください。チェックが入っていない場合、アクティブなビューに対してのみ変更が適用されます。全てのビューに変更を適用チェックボックスをクリックした後の変更だけが全てのビューに対して適用される点に注意してください。
- ツールバーを新たに追加: 新規ボタンをクリックして、表示されるツールバーの名前ダイアログにてツールバーの名前を入力します。作成されたツールバーにはコマンド998 タブからコマンドをドラッグすることができます。
- 追加されたツールバーの名前を変更: 追加されたツールバーをツールバーペインにて選択し、名前の変更ボタンをクリックします。表示されるツールバーの名前ダイアログにて名前の編集を行います。
- メニューバーのリセット: ツールバーペインにてメニューバーアイテムを選択し、リセットをクリックします。この操作によりアプリケーションがインストールされた初期の状態にメニューバーがリセットされます。
- ツールバーとメニューコマンドのリセット: 全てリセットボタンをクリックすることで、アプリケーションがインストールされた初期の状態に、全てのツールバーならびにメニューがリセットされます。
- ツールバーの削除: ツールバーペインにて削除するツールバーを選択し、削除ボタンをクリックします。
- 特定のツールバーにコマンドのテキストラベルを表示: 目的のツールバーを選択し、テキストラベルを表示チェックボックスにチェックを入れます。テキストラベルは各ビューに対して個別に有効化する必要がある点に注意してください。

24.14.10.3 ツール

ツール タブでは、XMLSpy 内部から外部アプリケーションを使用するコマンドをセット アップすることができます。これらのコマンドは、「ツール | ユーザー定義ツール」メニュー以下に表示されます。例えば、作成されたツールメニューのコマンドを「ツール | ユーザー定義ツール」以下からクリックすることで、XMLSpy のメインウィンドウに表示されているアクティブなファイルを、メモ帳のような外部アプリケーションにて開くことができます。



以下の操作により外部アプリケーションを使用するコマンドをセット アップすることができます:

1. メニューコンテンツペインにて、ペインのタイトル バーにある新規作成アイコンをクリックし、作成されたアイテムにメニューコマンドの名前を入力します。上のスクリーンショットでは、メモ帳で開くというメニューコマンドが作成されており、アクティブなドキュメントを外部アプリケーションの(W indows に搭載されている)メモ帳にて開くようなコマンドを作成します。新規作成アイコンをクリックすることで、更にコマンドを追加することもできます。アイテムを上へならびにアイテムを下へアイコンを使用することで、コマンドの相対的な位置を変更することができます。コマンドを削除するには、削除アイコンをクリックしてください。
2. 外部アプリケーションとコマンドを関連付けるには、メニューコンテンツペインにあるコマンドを選択し、コマンドフィールドにて外部アプリケーションの実行可能ファイルへの パスを入力するか、(入力フィールドの隣にあるボタンから)参照してください。上のスクリーンショットでは、メモ帳への パスがコマンドフィールドへ入力されています。
3. 引数フィールドにあるボタンをクリックすることで、外部アプリケーションに対して利用可能なアクションが表示されます(上のスクリーンショットを参照)。これらのアクションについては以下にあるリストを参照ください。アクションを選択すると、そのアクションに対応したコード文字列が引数フィールドに入力されます。
4. 処理を行う際の初期ディレクトリを指定する場合、その値を初期ディレクトリフィールドに入力してください。
5. 閉じるボタンをクリックすることで処理を完了します。

作成されたコマンドは、「ツール | ユーザー定義ツール」メニューと、プロジェクト ウィンドウにてファイルを右クリックすることで表示されるコンテキスト メニューの「ユーザー定義ツール」サブメニュー以下に表示されます。

新規作成されたコマンドをツール メニューにてクリックすると、コマンドに関連付けられたアクションが実行されます。上のスクリーンショットに示されたコマンドの例の場合、XMLSpy のメインウィンドウでアクティブなドキュメントがメモ帳で開かれます。外部アプリケーションコマンドへは、プロジェクト ウィンドウのユーザーツールコンテキスト メニューからもアクセスすることができます(プロジェクト ウィンドウにあるファイルを右クリックすることで

コンテキストメニューが表示されます)。プロジェクトウィンドウでは、複数のファイルを選択し、コンテキストメニューからコマンドを選択することで、複数のファイルに対して関連付けられたアクションを実行することができます。

引数

引数フィールドでは外部アプリケーションコマンドにより実行されるアクションが指定されます。以下の引数を利用することができます:

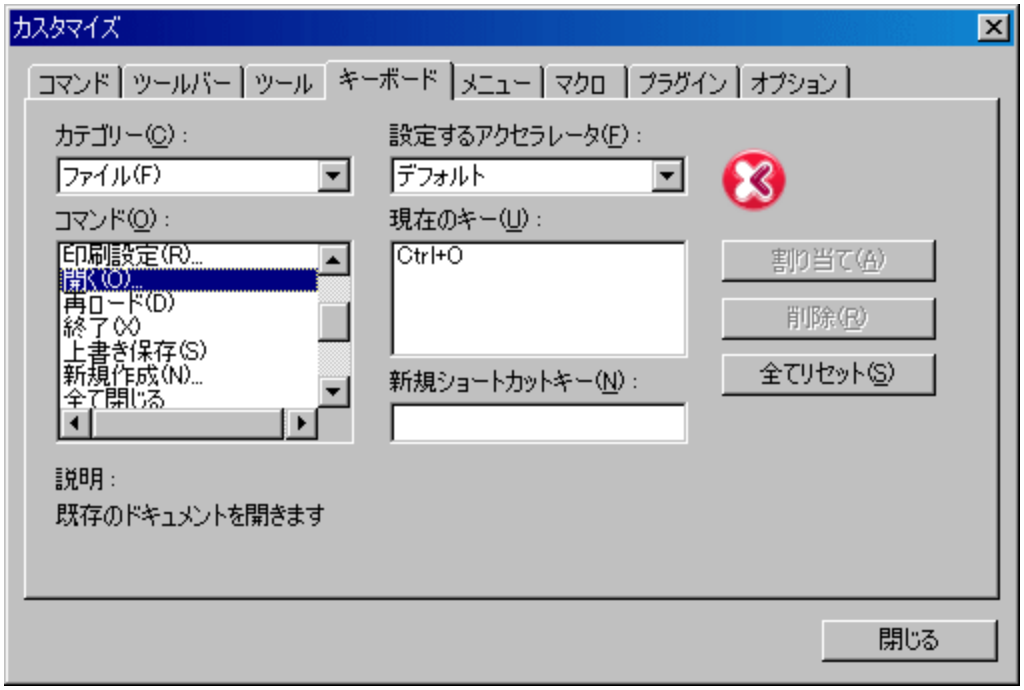
- アクティブなドキュメントのファイルパス: 「ユーザー定義ツール」メニューにあるコマンドから、XMLSpy にてアクティブになっているドキュメントを外部アプリケーションで開きます。プロジェクトウィンドウのコンテキストメニューからコマンドへアクセスした場合、選択されたファイルが外部アプリケーションで開かれます。
- プロジェクトファイルパス: XMLSpy のプロジェクトファイル(.spp ファイル)を外部アプリケーションにて開きます。

初期ディレクトリ

オプションとして、外部アプリケーションを実行する際にカレントディレクトリとして使用されるディレクトリへのパスを、**初期ディレクトリ**フィールドへ入力することができます。

24.14.10.4 キーボード

キーボードタブでは、キーボードショートカットを新たに作成したり、既存のショートカットを変更することができます。



以下の操作により、新たなショートカットをコマンドへ割り当てる、または既存のショートカットを変更することができます:

1. **カテゴリー**コンボボックスから**全てのコマンド**を選択します。関連付けられたコマンドとしてマクロが選択された[1008]場合、カテゴリーコンボボックスにマクロをセットすることもできます。
2. コマンドリストボックスにて、ショートカットを割り当てる、またはショートカットを変更するコマンドを選択します。
3. 新規ショートカットキーテキストボックスをクリックし、そのコマンドに対して割り当てるショートカットキーを押下します。入力されたショートカットが新規ショートカットキーテキストボックスに表示されます。そのショートカットがどのコマンドにも割り当てられていない場合、

割り当てボタンが選択可能状態になります。ショートカットが他のコマンドに既に割り当てられている場合、そのコマンド名がテキストボックス以下に表示され、割り当てボタンが選択不可能状態になります。新規ショートカットキーテキストボックスに入力された内容をクリアするには、Ctrl、Alt、またはShift のいずれかのキーを押下してください。

4. 割り当てボタンをクリックすることで、ショートカットの割り当てを行うことができます。ショートカットが現在のキーリストボックスに表示されます。複数のショートカットを同じコマンドへ割り当てることもできます。
5. 閉じるボタンをクリックすることで、変更を確定します。

ショートカットの削除

同じショートカットを複数のコマンドに対して割り当てることはできません。ショートカットを削除するには、現在のキーリストボックスにあるショートカットを選択し、削除ボタンをクリックし、閉じるボタンをクリックすることで削除を確定してください。

設定するアクセラレータ

現在割り当てられている機能はありません。

デフォルトのキーボードショートカット

インストール時に割り当てられているコマンドのリストを(機能ごとに)以下に示します: アプリケーションではキーボードマップダイアログ(「ヘルプ | キーボードマップ 1048」)では、ショートカットとその説明とともに、コマンドのリストを確認することができます。

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

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

ファイルとアプリケーションコマンド

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

その他のキー

(C) 2015-2021 Altova GmbH

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

編集コマンド

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

テキストビューコマンド

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

グリッドビューコマンド

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

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

スキーマビューコマンド

CTRL + 要素をダブルクリック	要素の定義の表示

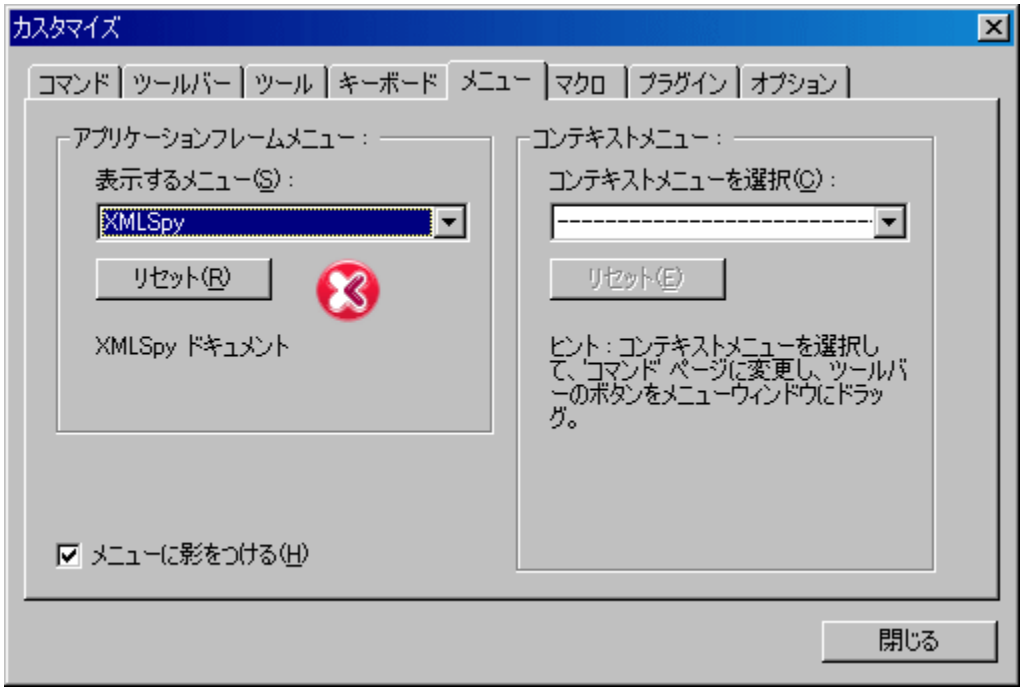
デバッガーコマンド

F9	ブレークポイントの挿入/削除
F9 + Shift	トレースポイントの挿入/削除
F9 + CTRL	ブレークポイントの有効化/無効化
F9 + Shift + CTRL	トレースポイントの有効化/無効化
F11	ステップイン
F11 + Shift	ステップアウト
F11 + CTRL	ステップオーバー
F11 + Alt	デバッガーを開始する

24.14.10.5 メニュー

メニュータブでは、デフォルトならびにアプリケーションメニューバーという2つのメインメニューバー、そしてアプリケーションのコンテキストメニューをカスタマイズすることができます。

(C) 2015-2021 Altova GmbH



デフォルトメニューバーとアプリケーションメニューバーのカスタマイズ

デフォルトのメニューバーとは、メインウィンドウにてドキュメントが開かれていない状態で表示されるメニューバーのことです。アプリケーションメニューバーとは、1つ以上のドキュメントがメインウィンドウにて開かれている状態で表示されるメニューバーのことです。各メニューバーは、お互いに干渉しあいかたちで個別にカスタマイズすることができます。

メニューバーをカスタマイズするには、「**表示するメニュー**」コンボボックスから目的のメニューを選択し、カスタマイズダイアログのコマンドタブ 998 へ切り替えます。その後コマンドリストボックスに表示されたコマンドを、メニューバーまたはメニューへドラッグします。

コマンドの削除とメニューバーのリセット

メニュー内のメニュー全体またはコマンドを[**削除**]するには以下を行います:

1. アプリケーションフレームメニューペイン内で *デフォルト*（ドキュメントが開かれていない場合）または（1つ以上のドキュメントが開かれている場合使用可能なメニューを表示する）*XMLSpy* を選択します。
2. 開かれているカスタマイズダイアログから（i）アプリケーションのメニューバーから削除するコマンド（ii）これらのメニューから削除するコマンドを選択します。
3. （i）メニューバーからメニューをドラッグ、または、メニューからメニューコマンドをドラッグ、または（ii）メニューまたはメニューコマンドを右クリックして[**削除**]を選択します。

「*表示するメニュー*」コンボボックスにてメニューを選択し、「リセット」ボタンをクリックすることで、2つのメニューバー（デフォルトならびにアプリケーションメニューバー）のそれぞれをリセットすることができます。

アプリケーションのコンテキストメニューをカスタマイズする

コンテキストメニューとは、アプリケーションインターフェースに表示されたオブジェクトを右クリックすることで表示されるメニューのことです。以下の方法により、コンテキストメニューのカスタマイズを行うことができます:

1. コンテキストメニューを選択コンボボックスにて目的のコンテキストメニューを選択します。コンテキストメニューが表示されます。

2. カスタマイズダイアログのコマンドタブ[998] へ切り替えます。
3. コマンドリストボックスからコンテキストメニューへコマンドをドラッグします。
4. コンテキストメニューからコマンドを削除するには、コンテキストメニューに表示されたコマンドを右クリックして、「削除」を選択します。コンテキストメニューからコマンドをドラッグすることでも、そのコマンドを削除することができます。

コンテキストメニューを選択コンボボックスにてメニューを選択し、「**リセット**」ボタンをクリックすることで、コンテキストメニューをオリジナルの状態にリセットすることができます。

メニューに影を付ける

「**メニューに影を付ける**」チェックボックスを有効にすることで、全てのメニューに影が与えられます。

24.14.10.6 マクロ

マクロタブでは、XMLSpy のスクリプトエディターにより作成されたマクロに対するアプリケーションコマンドを作成することができます。関連付けられたマクロを実行するこれらアプリケーションコマンドは、マクロタブから直接、またはカスタマイズダイアログのコマンドタブ[998] を使用することにより、メニューやツールバーにて利用することができるようになります。アプリケーションコマンドとして、カスタマイズダイアログのキーボードタブ[1003] にてショートカットを割り当てることもできます。

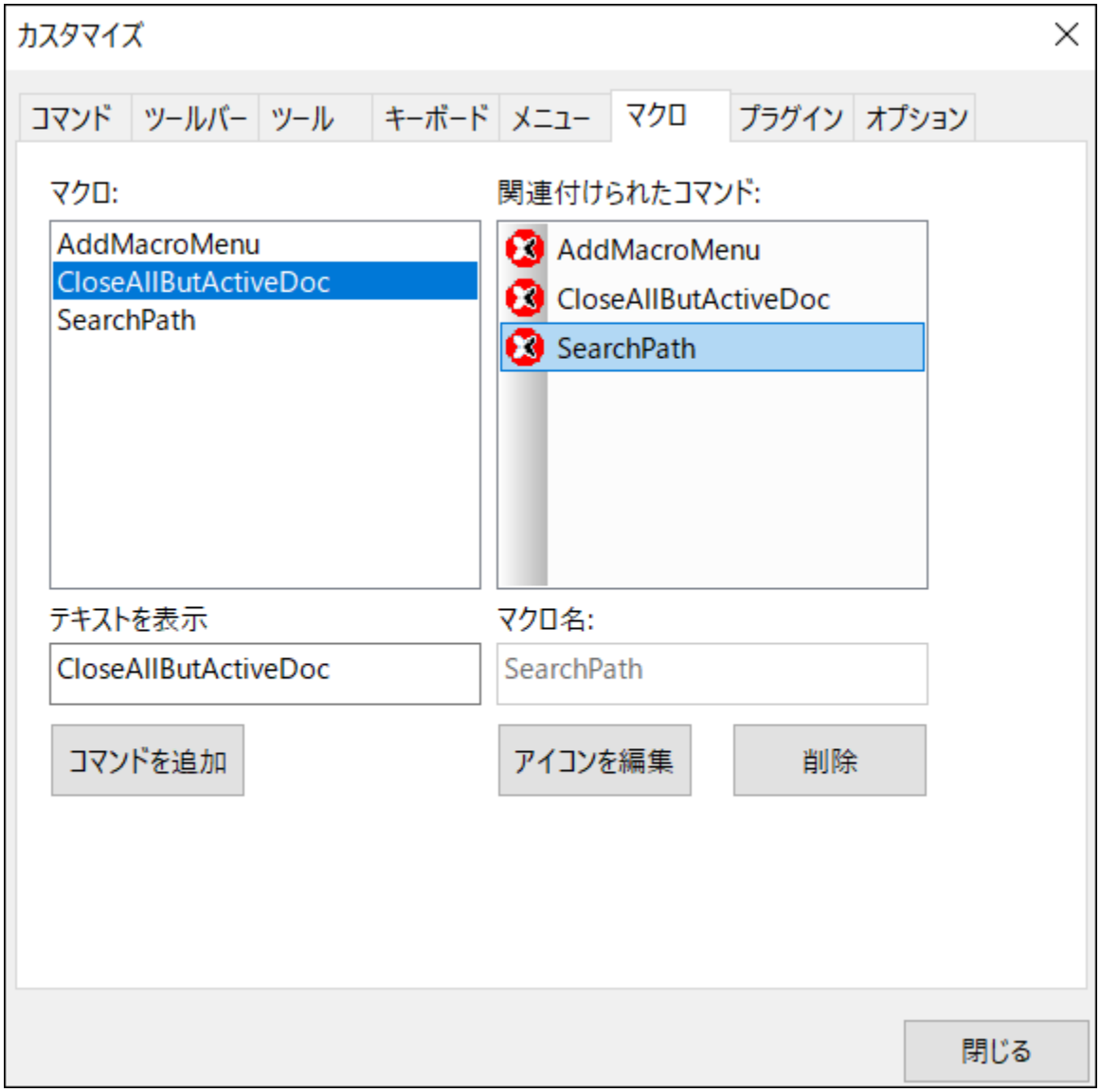
XMLSpy におけるマクロの作動

XMLSpy におけるマクロは以下のように動作します:

- XMLSpy のスクリプトエディター[1056] により、Altova スクリプトプロジェクト（.asprj ファイル）が作成されます。スクリプトプロジェクトには、XMLSpy で使用することのできるマクロを含めることができます。
- XMLSpy では、(i) オプションダイアログのスクリプトタブ[1040] にて指定されるアプリケーションプロジェクトと、(ii) スクリプト設定ダイアログ[808] (プロジェクト｜スクリプト設定[808])内で指定される アクティブな XMLSpy プロジェクト[676] スクリプトロジェクト。
- これら2つのスクリプトプロジェクト内にあるマクロは「ツール｜ マクロ」メニューから利用することができ、カスタマイズダイアログのマクロタブではマクロをアプリケーションコマンドとしてセットすることができます（以下のスクリーンショットを参照）。マクロをアプリケーションコマンドとしてセットすると、メニューやツールバーにそのコマンドを配置することができるようになります。

マクロに対してアプリケーションコマンドを作成する

スクリプトエディター[1056] (ツール｜スクリプトエディター[1056]」)にて目的のマクロを作成し、スクリプトプロジェクトとして保存します。このファイルを(オプションダイアログのスクリプトタブ[1040] から) アプリケーションスクリプトプロジェクトとして、または(アプリケーションプロジェクトの スクリプト設定ダイアログ[808] (プロジェクト｜スクリプト設定[808]」) から) アクティブなアプリケーションプロジェクトのスクリプトプロジェクトとして指定することができます。スクリプト内部にあるマクロがマクロタブの「**マクロ**」ペインに表示されます（以下のスクリーンショットを参照）。
マクロに対するアプリケーションコマンドを作成するには、「**マクロ**」ペインにてマクロを選択し、「**テキストを表示**」テキストボックスにてコマンドのテキストを入力した後に、「**コマンドを追加**」ボタンをクリックします（以下のスクリーンショットを参照）。選択されたマクロに関連付けられたコマンドが、「**関連付けられたコマンド**」リストボックスに追加されます。



関連付けられたコマンドのアイコンを編集するには、そのコマンドが選択された状態で**アイコンを編集**ボタンをクリックします。**削除**ボタンをクリックすることで、関連付けられたコマンドを削除することができます。

マクロに関連付けられたコマンドをメニューやツールバーに配置する

マクロに関連付けられたコマンドをメニューやツールバーに配置するには、2つの方法があります:

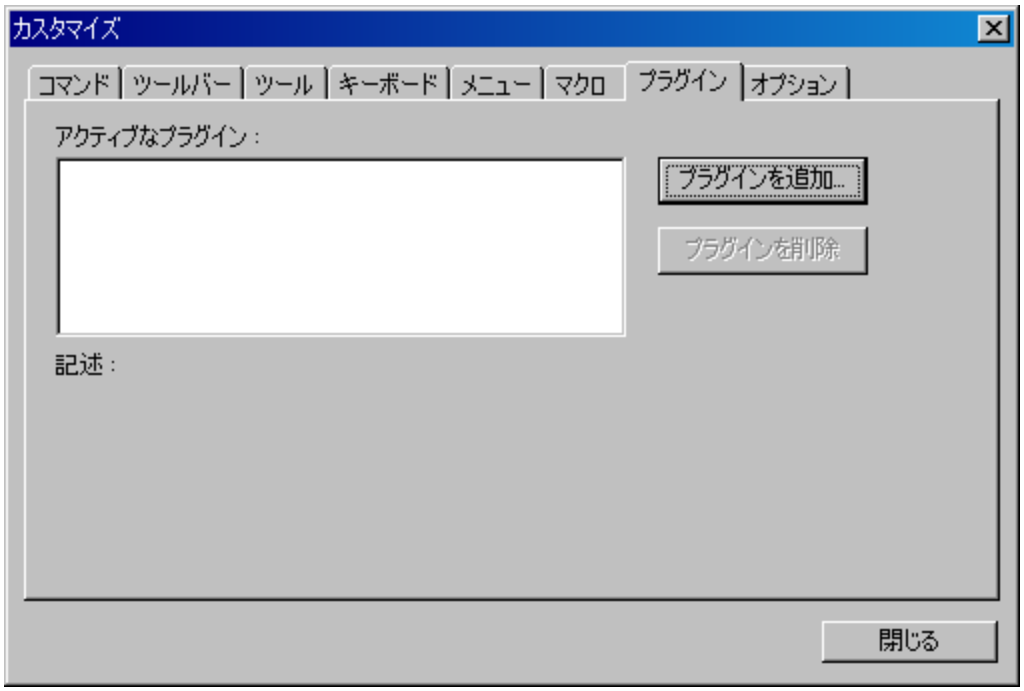
- 関連付けられたコマンドリストボックスから、メニューやツールバー上にある目的の場所へコマンドをドラッグする。
- カスタマイズダイアログのコマンドタブ998に用意されているメカニズムを使用する。

どちらの場合でも、指定された場所にコマンドが作成されます。メニューまたはツールバー上にあるコマンドをクリックすることで、関連付けられたマクロが実行されます。

メモ: 関連付けられたコマンドとしてマクロがセットされた場合、キーボードのショートカット[1003]を割り当てることができます。カスタマイズダイアログのキーボードタブ[1003]にて、カテゴリーコンボボックスから**マクロ**を選択し、目的のマクロを選択した状態でショートカットをセットしてください。

24.14.10.7 プラグイン

「**プラグイン**」タブでは、アプリケーションやツールバー動作するようプログラムされたプラグインの統合とコマンドの配置を行うことができます。(*下のスクリーンショット*)プラグインタブでは、「**プラグインを追加**」をクリックすることで、プラグインのDLLファイルを参照することができます(*下の「プラグインの作成」を参照してください*)。「OK」をクリックすることでプラグインが追加されます。アプリケーションには複数のプラグインを追加することができます。



プラグインの追加に成功すると、プラグインの説明がダイアログに表示され、**プラグインを削除**ボタンが選択可能になります。プラグインコードによりツールバーやメニューが作成されると、それらがアプリケーションインターフェースに表示されます。プラグインを削除するには、**プラグインを削除**ボタンをクリックしてください。

プラグインの作成

サンプルプラグインのソースコードはアプリケーションのマイドキュメント フォルダー[27]マイドキュメント フォルダー: `Examples\IDEPlugin` フォルダー内に収められています。ソースコードからプラグインを作成するには以下を行います:

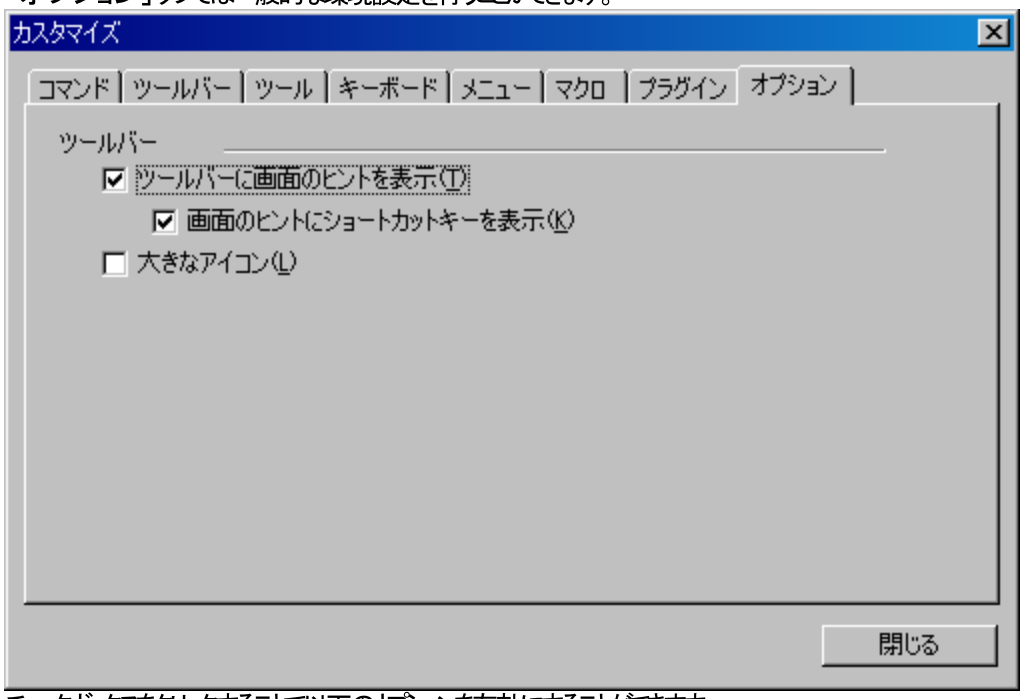
1. Visual Studio 内でプラグインとして作成するソリューションを開きます。
2. ビルドメニュー内のコマンドを使用してプラグインを作成します。
3. プラグインのDLLファイルは`Bin` または`Debug` フォルダー内で作成されます。このDLLファイルは、プラグインとして追加される必要があるファイルです(*上記参照*)。

プラグインの作成に関する詳細については、IDE プラグイン[1081]のセクションを参照ください。

(C) 2015-2021 Altova GmbH

24.14.10.8 オプション

「オプション」タブでは一般的な環境設定を行うことができます。

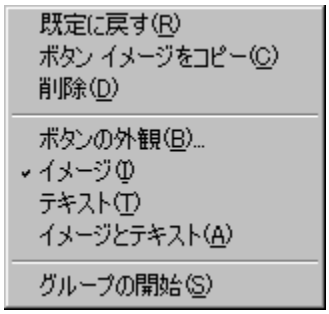


チェックボックスをクリックすることで以下のオプションを有効にすることができます:

- ツールバーに画面のヒントを表示: ツールバーのアイコン上にマウスポインターが配置されると、ポップアップが表示されます。ポップアップにはアイコン機能の説明と、(ショートカットキーを表示オプションが有効になっていれば) 関連付けられたショートカットが表示されます。
- 画面のヒントにショートカットキーを表示: 画面のヒントにショートカットキーに関する情報を表示するか指定することができます。
- 大きなアイコン: ツールバーアイコンの大きさを通常のサイズと大きめのサイズへ切り替えることができます。

24.14.10.9 コンテキストメニューのカスタマイズ

コンテキストメニューのカスタマイズコマンドは、カスタマイズダイアログが表示されている状態で、メニュー、メニューコマンド、またはツールバーアイコンを右クリックすることにより表示されます。

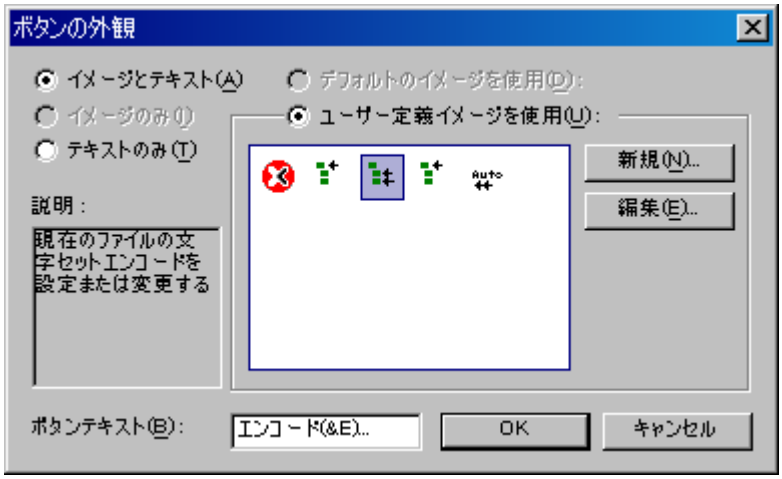


以下の機能を利用することができます:

- 規定に戻す: 現在機能は与えられていません。
- ボタンイメージをコピー: 右クリックしたアイコンをクリップボードにコピーします。
- 削除: 選択されたメニュー、メニューコマンド、またはツール バーアイコンを削除します。削除されたアイコンを再度表示する方法については、下記を参照ください。
- ボタンの外観: ボタンの外観ダイアログが表示され、選択されたツール バーアイコンの外観を定義するためのプロパティをセットすることができます。詳細については以下の記述を参照ください。
- イメージ、テキスト、イメージとテキスト: 相互に排他的なオプションで、選択されたツール バーアイコンをアイコンだけ、テキストだけ、またはアイコンとテキストの両方で表示するかを指定します。いずれかのオプションを選択することで、必要な変更が行われます。
- グループの開始: 選択されたツール バーアイコンの左側にグループを分離する垂直の線を挿入します。これにより選択されたツール バーアイコンがグループ最初のアイコンとなります。

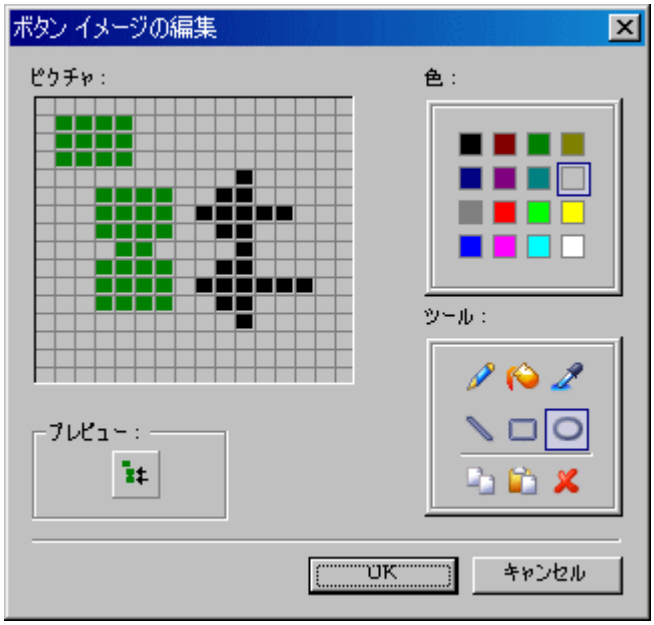
ボタンの外観ダイアログ

ツール バーアイコンを右クリックして、「**ボタンの外観**」を選択することで、ボタンの外観ダイアログが表示されます(以下のスクリーンショットを参照)。このダイアログでは、ツール バーアイコンのイメージやテキストを選択することができます。現在このダイアログでは、マクロならびにプラグインのツール バーアイコンだけを編集することができます。



選択されたツール バーのアイコン(**コンテキストメニューのカスタマイズ**にて右クリックされたアイコン)に対して以下の編集機能を使用することができます。

- イメージのみ、テキストのみ、イメージとテキスト: 目的のラジオボタンを選択して、ツール バーのアイコンが表示される形式を指定することができます。
- イメージの編集: イメージのみまたはイメージとテキストが選択された場合、イメージの編集オプションが有効になります。「新規」ボタンをクリックすることでユーザー定義の新たなイメージが作成され、イメージペインに追加されます。イメージを選択して、「編集」ボタンにより編集を行うことができます。



- イメージ選択: イメージペインから目的のイメージを選択し、OK をクリックすることにより選択されたイメージをツールバーで使用することができます。
- テキストの編集とセレクター: テキストのみまたはイメージとテキストが選択された場合、ボタンテキストテキストボックスが選択可能になります。テキストを入力または編集し、OK をクリックすることで、ツールバーアイコンのテキストを変更することができます。

メモ ボタンの外観ダイアログでは、メニューコマンドのテキストを編集することもできます。カスタマイズダイアログが開かれている状態でメニューコマンドを右クリックして、「**ボタンの外観**」を選択し、「**ボタンテキスト**」テキストボックスにてメニューコマンドのテキストを編集してください。

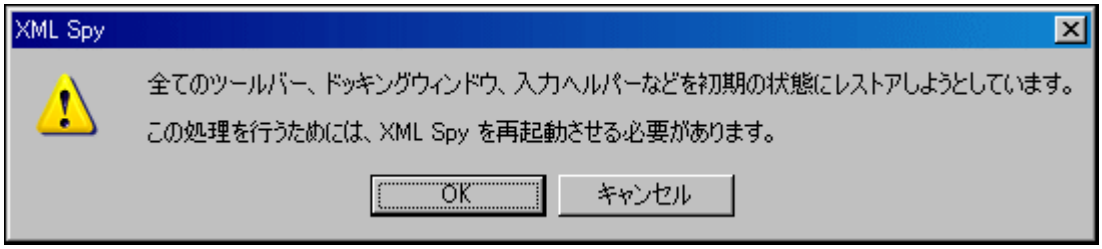
削除されたメニュー、メニューコマンド、ツールバーコマンドを復元

コンテキストメニューのカスタマイズにある**削除**コマンドにより、メニューやメニューコマンド、ツールバーアイコンが削除された場合、以下の方法により復元することができます:

- メニュー: メニューオプションから「ツール| カスタマイズ| メニュー 1006 」を選択し、アプリケーションフレームメニューペインにあるリセットボタンをクリックしてください。他にも「ツール| カスタマイズ| ツールバー 1000 」を選択し、ダイアログ左側でメニューバーを選択した状態でリセットボタンをクリックすることもできます。
- メニューコマンド: メニューオプションから「ツール| カスタマイズ| コマンド 998 」を選択し、コマンドリストボックスから目的のコマンドをメニューへドラッグしてください。.
- ツールバーアイコン: メニューオプションから「 ツール| カスタマイズ| コマンド 998 」を選択し、コマンドリストボックスから目的のコマンドをツールバーへドラッグしてください。

24.14.11 ツールバーとウィンドウの復元

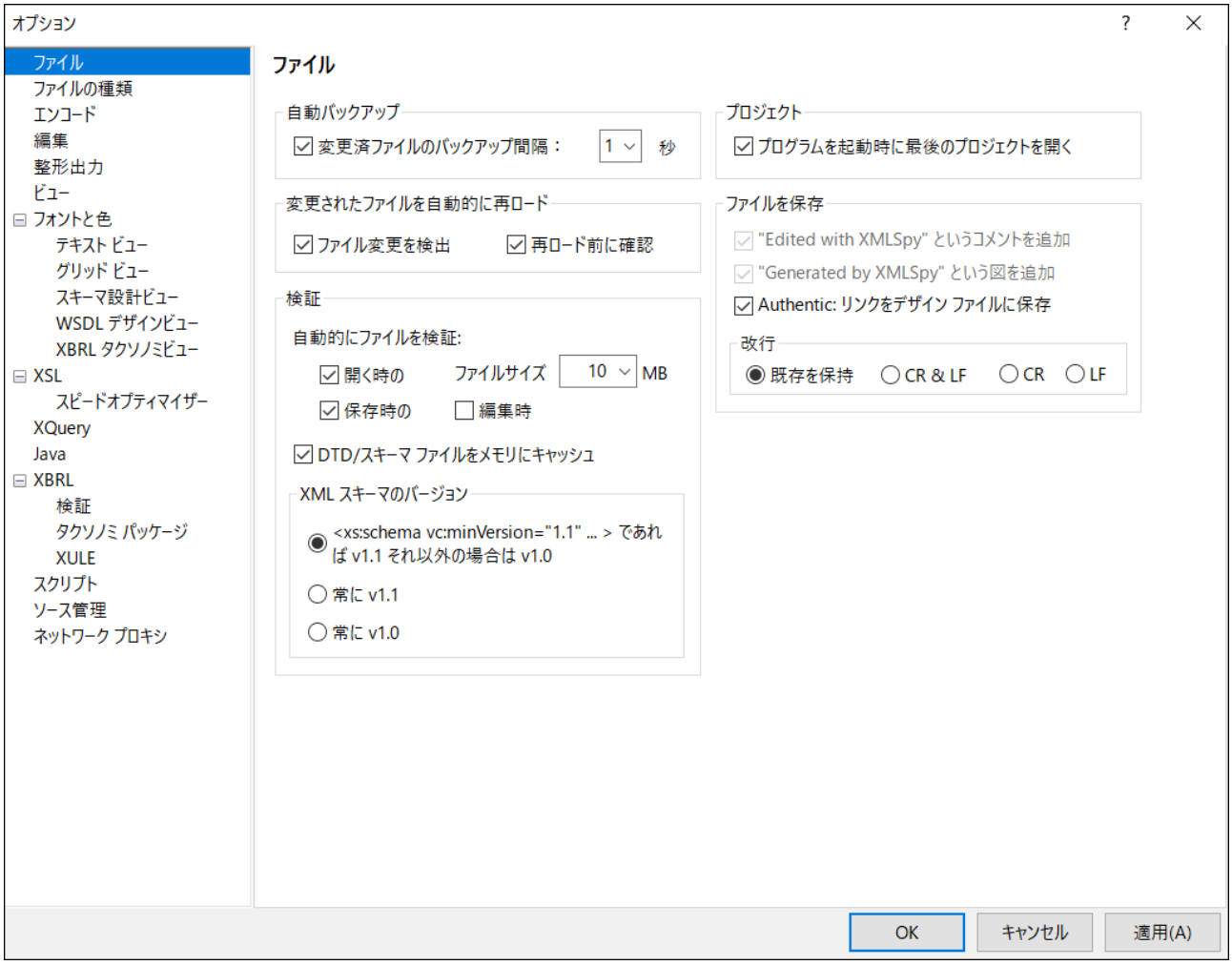
ツールバーとウィンドウの復元コマンドにより、XMLSpy をデフォルト設定の状態で再起動することができます。終了する前に XMLSpy の終了を確認するダイアログが表示されます（以下のスクリーンショットを参照）。



このコマンドは、ツールバーやウィンドウのサイズ変更、移動、または非表示にするように指定した後、ツールバーやウィンドウをオリジナルの状態に戻したい場合に使用することができます。

24.14.12　オプション

[ツール | オプション] コマンドによりグローバルなアプリケーション設定を定義することができます。これらの設定は、セクションにより整理されています（*下のスクリーンショットXBRL オプションは Enterprise Edition のみでご使用いただけます内の左側のペイン参照*）。例えば、ファイルセクション 1015（*下のスクリーンショットを参照*）には、XMLSpy によるファイルの開き方、及び、保存の方法を指定することのできるオプションが含まれています。特定のセクションのオプションを指定するには、左側のペイン内のそのセクションを選択し、必要とするプロパティの値を指定します。「**OK**」ボタンにより、レジストリへの変更が保存され、ダイアログが閉じられます。「**適用**」ボタンにより現在開かれているドキュメントへの変更を表示することができます。



オプションダイアログの各セクション詳細については、このセクションのサブセクションを参照ください。

24.14.12.1 ファイル

ファイルセクションでは、XMLSpy にてファイルを開く、または保存する際の振る舞いを定義することができます。エンコードセクション[1020]にも関連する設定が存在します。

自動的なバックアップ

このオプションが有効化されていると現在編集中のファイルは自動的にバックアップされます。5 秒から 60 秒までの間のバックアップの間隔をコンボボックスから選択、または 300 秒までのカスタム値を入力することができます。詳細に関しては、ファイルの自動的なバックアップ 120 のセクションを参照してください。

変更されたファイルの自動的な再ロード

マルチユーザー環境内で作業する場合、または、サーバー上で動的に生成されるファイルと作業する場合、インターフェイス内で現在開かれているファイルへの変更を確認することができます。XMLSpy が開かれているドキュメント内の変更を検知する都度、変更されたファイルを再ロードするかプロンプトされます。

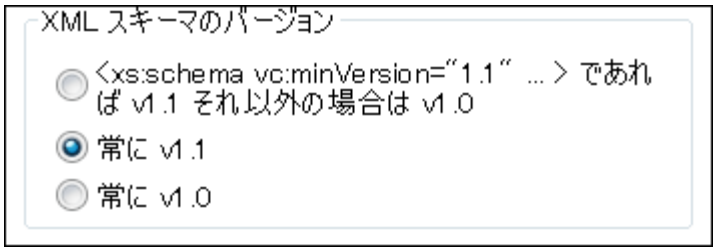
検証

DTD やスキーマにより XML ドキュメントの構造を定義している場合、ドキュメントを開くまたは保存する際に検証を行うかどうか指定することができます。自動的に行われる検証は、ドキュメントの上限を定めることで、効果的な編集作業を行うことができます。ドキュメントが妥当でない場合、エラーメッセージが表示されます。ドキュメントが妥当な場合は、メッセージが出力されることなく、処理が行われます。XMLSpy ではこれらのファイルをメモリ上にキャッシュすることで、（例えば、参照されているスキーマが URL によりアクセスされた場合など）不必要な再ロードを抑えることができます。スキーマ位置の宣言に URL を使用している場合、「DTD/スキーマファイルをメモリにキャッシュ」オプションを無効にすることで、スキーマに対して行われた変更を、キャッシュを介さず直接保存することができます。

編集中 オプションが選択されているとテキストビュー[122]とJSON グリッドビュー[485]での入力中に検証が実行されます。このオプションはデフォルトで選択されています。詳細に関してはテキストビュー内でのXML 検証[271]を参照してください。

XML スキーマバージョン

スキーマビュー内で有効化されたXSD モードは以下により異なります(i) 属性の存在/不在—存在する場合、XSD ドキュメントの`/xs:schema/@vc:minVersion` 属性 の値(ii) オプションダイアログのファイルセクション(「**ツール | オプション**」内で選択されるXML スキーマバージョンのオプション。*してのスクリーンショット*)。



以下のシチュエーションが可能になります。下のテーブル内 *XML スキーマバージョン*は、上に示されるXML スキーマバージョン ペイン内の選択を参照しています。テーブル内の`vc:minVersion` 値は、XML スキーマドキュメント内の`xs:schema/@vc:minVersion` 属性の値を参照しています。詳細に関しては、編集ビュー | スキーマビュー | XSDMode [153]のセクションを参照してください。

XML スキーマバージョン	`vc:minVersion` 属性	XSD モード
常にv1.0	不在、または、存在していても値を持っていない	1.0
常にv1.1	不在、または、存在していても値を持っていない	1.1
@ vc:minVersion の値	属性は、`1.1` の値を持っている	1.1
@ vc:minVersion の値	属性が不在、または、存在していても値が`1.1`以外	1.0

プロジェクト

XMLSpy を起動すると、最後に使用されたプロジェクトが自動的に開くように指定することができます。

ファイルを保存

XML ドキュメントを保存する際、XMLSpy により、`<!-- Edited with XMLSpy http://www.altova.com -->` というコメントが、ドキュメントの上部に挿入されます。このオプションを無効にするにはユーザーのライセンス登録が必要で、拡張グリッドビューまたはスキーマデザインビューにて編集を行い、ファイルを保存した際に有効になります。

(「**スキーマ設計 | ドキュメンテーションの生成**」により)コンテンツモデルダイアグラムを保存する際、XMLSpy によりXMLSpy のロゴが挿入されます。このオプションを無効にするには、ライセンスの登録が必要になります。

XML ファイルにStyleVision Power Stylesheet が関連付けられている場合、「Authentic : リンクをデザインファイルに保存」オプションを有効にすることで、XML とともにStyleVision Power Stylesheet を保存することができます。

改行

既存を保持が選択されていると、ファイルを開いたときに、ドキュメント内にあった改行コード文字がそのまま保持されます。または、3つのコード内のコードの変更を選択することができます: **CR&LF** (PC)、**CR** (MacOS)、または**LF** (Unix)。

保存して完了

設定が完了したら、「OK」をクリックして完了します。

24.14.12.2 ファイルタイプ

ファイルタイプセクションでは、各種ファイルに対するXMLSpy の振る舞いを指定することができます。*（下のスクリーンショットでは、XBRL とAvro へのサポートに関するオプションの一部は、Enterprise Edition のみで使用が可能なことに注意してください）。*

ファイルタイプリストボックスからファイルの種類を選択し、その種類のファイルに対する動作をカスタマイズすることができます。

Windows エクスプローラー設定

Windows エクスプローラーで使用されるファイルタイプの説明と、MIMIE 準拠のコンテンツタイプを定義するほか、この種類のドキュメントを編集するデフォルトディターとしてXMLSpy を指定することができます。

準拠

XMLSpy では、様々な種類のファイルに対して使用することのできる数多くの機能が搭載されています。特定のファイルタイプに対して使用されるオプションの種類は、この準拠オプションによりセットされます。XMLSpy ではファイルタイプをXML, XQuery 332、ZIP 566、JSON 462 そしてその他のテキスト形式に準拠するよう設定することができるほか、XML に準拠したファイルも、XML、DTD、そしてXML エンティティファイルタイプと細分化することができます。ファイルの型の大半はファイル型に適切なデフォルトの準拠により定義されています。新規のファイル型を追加、または、準拠の他の型にファイルを設定することを希望する場合を除いてこれらの設定を変更しないことが奨励されます。

(C) 2015-2021 Altova GmbH

デフォルトビュー

このグループでは、各ファイルの種類に対して使用されるデフォルトのビューを指定することができます。上のスクリーンショットは、エディションのファイル型のセクションを表示しています。使用中のエディションがEnterprise エディションでは無い場合、スクリーンショットに表示されているより少ないビューの種類のみが搭載されています。特定の準拠が特定のビューでのみ確認できる場合、ビューはデフォルトで選択されており、ビューセクションは無効化されています。例えば、XQuery Update ドキュメントはテキストビューでのみ確認することができ、このビューはデフォルトで選択されており、ビューセクションは無効化されています。。

グリッドビュー

このチェックボックスにより、グリッドビューで自動的にテーブル表示が使用されるか指定することができます。

テキストビュー

このチェックボックスにより、特定の種類のファイルに対して、構文の色分けを行うよう指定することができます。

自動検証を無効化

このオプションにより、各ファイルに対して自動検証を有効にすることができます。自動検証は通常、ファイルが開かれたとき、またはビューが切り替えられたときに行われます。

空の要素を省略して<E/> で保存する

XML ドキュメントや、XML ドキュメントにより出力されたドキュメントを使用するアプリケーションの中には、XML 1.0 仕様により定義されている(<Element/> という形式の)空要素が原因でエラーを起こすことがあります。このオプションを無効にすることで、XMLSpy は、<Element></Element> という形式で要素の保存を行います。

RaptorXML Server を使用して開く/保存時に検証する

ファイが開かれて保存される際に選択されたファイル型のファイルを検証するためにRaptorXML Server が使用されるかを指定します。
RaptorXML Server はセットアップされ、構成されている必要があります。994

新たなファイル拡張子を追加

新しいファイルタイプを、ファイルタイプリストに追加します。ファイルタイプを追加した後には、このタブにあるオプションを使用することで、このファイルタイプの定義を行う必要があります。

選択されたファイル拡張子を削除

現在選択されているファイルタイプと、全ての関連する設定を削除します。

保存して完了する

設定が完了したら、「OK」をクリックして完了します。

24.14.12.3 エンコード

エンコードセクションでは、ファイルのエンコーディングを指定することができます。

新規 XML ファイルのデフォルトエンコーディング

ドロップダウンリストから新規 XML ファイルで使用されるデフォルトエンコーディングを選択することができます。ここで指定したエンコーディングが、新たなドキュメントを作成する際、XML 宣言にて使用されます。デフォルトエンコーディングに2バイトまたは4バイトのエンコーディング(UTF-16、UCS-2 またはUCS-4)をデフォルトエンコーディングとして選択した場合、リトルエンディアン、またはビッグエンディアンを選択することができます。

既存のXML ファイルにおけるエンコーディングはそのまま保持されますが、「 ファイル| エンコード 751 」コマンドにより変更することができます。

XML ファイルを未知のエンコードを使用して開く

XML ファイルのエンコーディングを決定することができない、またはXML ドキュメントにてエンコーディングの指定がなされていない場合、このコンボボックスで選択したエンコーディングがファイルを開く際に使用されます。.

XML でないファイルを開く際のエンコーディング

既存の、そして新規に作成されたXML 以外のファイルは、このコンボボックスで選択したエンコーディングにより開かれます。ドキュメントのエンコーディングは、「 ファイル| エンコード 751 」コマンドにより変更することができます。

BOM (バイト・オーダー・マーク)

2バイトまたは4バイトのエンコーディングによりドキュメントを保存する場合、ドキュメントの保存を、(i) リトルエンディアンオーダー、またはリトルエンディアンBOM により行うか(UTF-8 でない場合、常に BOM を作成)、(ii) 検出されたバイトオーダーまたは検出された BOM により行うか(保存時に検出した BOM を保持)指定することができます。

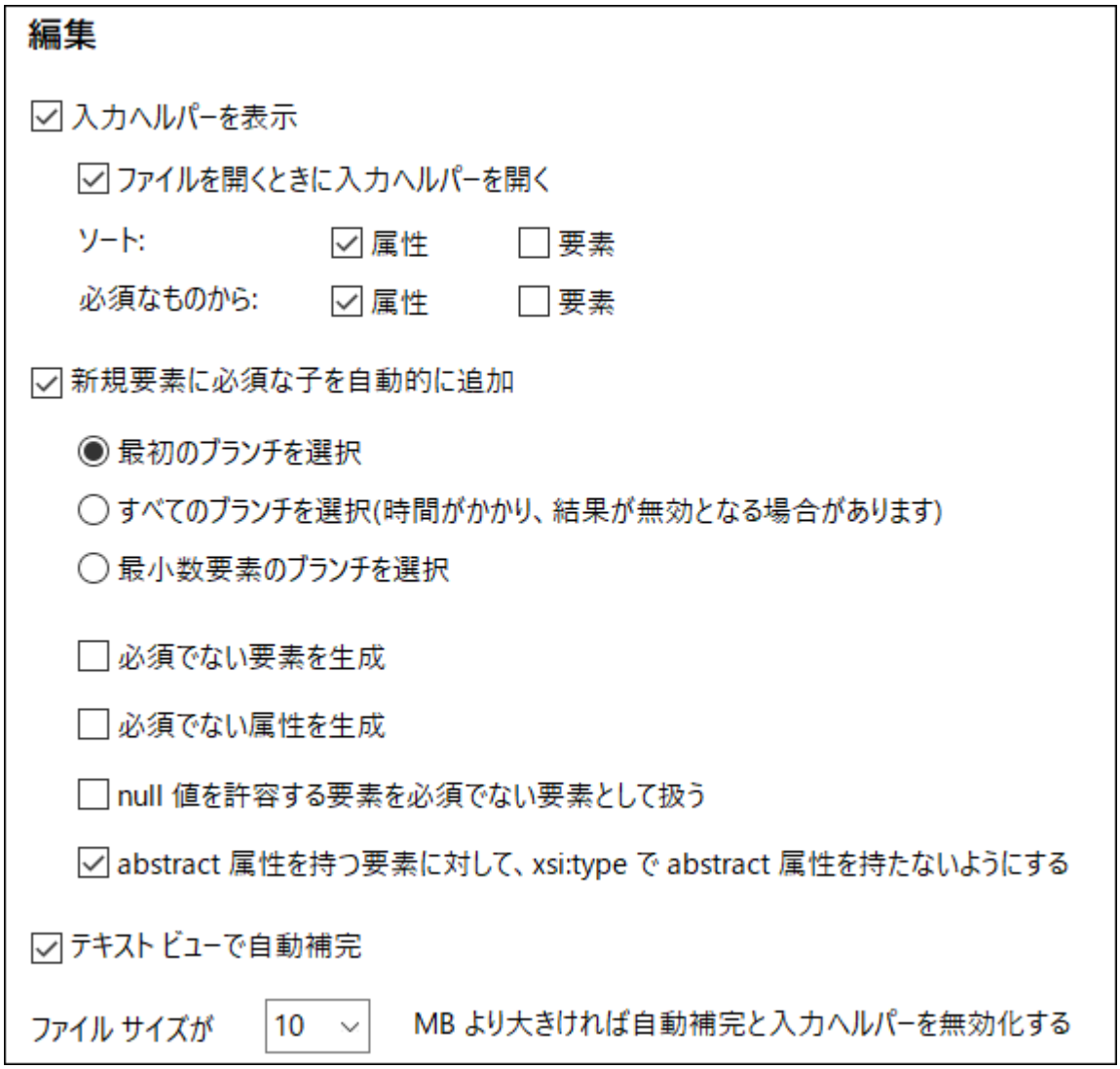
保存して完了する

「OK」をクリックして完了してください。

(C) 2015-2021 Altova GmbH

24.14.12.4 編集

編集セクションでは、XMLSpy の編集における振る舞いを指定することができます。



入力ヘルパー

ドキュメントの編集中 XMLSpy はこれらの設定をベースにしてインテリジェントな編集機能を提供します。このペイン内で入力ヘルパー内に表示されるアイテムの順序などを含む入力ヘルパーの振る舞いの多種のアスペクトをカスタム化することができます。編集中のファイル型に適切な場合にここで行われるカスタム化の設定は適用されます。例えば、ファイルを開く際に入力ヘルパーをロードし属性を置き換えるオプションは DTD または XQuery ドキュメントには適用することができません。

XML 構造を XML スキーマから作成する

XML スキーマをベースにする新規の XML ドキュメントを作成する場合、スキーマ内の定義から派生した構造を持つドキュメントが生成されます。下で説明されている設定はこの構造の作成に関連する抽象的なアスペクトのいくつかを決定します。

必須の子を自動的に付加する
スキーマ内の `choice` グループの必須の子要素はこのペイン内での設定をベースにして自動的に付加されます。(i) `choice` グループの最初のブランチ(要素) (ii) 全てのブランチ、または(iii) 子孫要素の最小数を持つブランチが生成されます。`choice` グループから1つのブランチのみが許可されているためNote that the *全てのブランチ* の選択は無効なドキュメントを生成する場合があることに注意してください。

抽象型の必須ではないノードと要素
必須ではない要素または属性を追加するには、対応するオプションを選択してください。これらのオプションが選択されていない場合、必須のノードのみが追加されます。また(i) nillable 要素の要素コンテンツを必須では内に設定し(ii) 抽象型の要素の `xsi:type` として必須ではない型を使用します。o

テキストビュー

*自動補完*オプションにより、より曖昧性の少ない構造のコンポーネントを作成することができます。例えば、要素の開始タグを閉じる山括弧が入力されると、その要素の終了タグも自動的に追加されます。

テキストビューでは *自動補完の無効化* コンボボックス内で指定されているよりファイルが大きい場合自動補完や入力ヘルパーを無効にすることができます。大きなファイルに対する編集のスピードを向上させる場合この機能は役に立ち、自動補完機能と入力ヘルパーを使用しないで行うことができます。このオプションのために指定されているサイズよりもファイルサイズが大きい場合、テキストビューコンテキストメニューには自動補完と入力ヘルパーの使用をオンとオフに切り替えるためのトグルコマンドが含まれます。(このオプションのために指定されているサイズよりも大きいファイルの場合)編集エイドを編集中にオンとオフいつでも切り替えることができます。このオプションのために指定されている値が開かれているファイルのサイズよりも小さい場合、エラーメッセージ内で指定されているロケーションはテキストビュー内のロケーションに正確に対応しません。

保存して完了

設定が完了したら、「OK」をクリックして完了します。

24.14.12.5 整形出力

整形出力 *セクション(下のスクリーンショット参照)* によりテキストが テキストビューこのセクションの定義は以下のカテゴリに整理されています:

- XML 設定(これらの設定にアクセスするためにXML タブを選択します。 *下の左側のスクリーンショット*)
- JSON 設定(これらの設定にアクセスするためにJSON タブを選択します。 *下の右側のスクリーンショット*)
- テキストビュー設定(設定ダイアログにアクセスするためにボタンをクリックします)

(C) 2015-2021 Altova GmbH

整形出力

テキスト ビューにてコマンドが実行される、他のすべてのビューから切り替えや保存が行われたときに整形出力が使用されます。

テキストビューのタブ構成によりインデントは決定されます => テキスト ビュー 設定...

XML JSON

空の要素: ◉自己終了 ○スペースのある自己終了 ○終了タグ ○新規のライン内の終了タグ

インライン属性: ◉常に ○最大 5 属性 ○なし

属性値: ☐スペース ☐単一引用符付き

空白の保持: pre, code, script ...

```
<root attribute="value">
    <empty/>
    <element a1="val1" a2="val2" a3="val3"/>
    <pre>
        <span>whitespace</span> <b>preserve</b>
</pre>
</root>
```

XML 設定

以下の設定を使用することができます:

- どのように要素が作成されドキュメント内で表示されるか
- インライン属性は親要素として同じラインまたは新規のラインに表示されます。
- (等号記号の両端に空白が存在するか、および値が単一または二重引用符で囲まれているかなど)属性値の表記方法
- 空白として保持する要素の設定
- テキストビュー設定 969 ダイアログ内で指定されているインデントが(ペインの上のチェックボックスを介して指定され)使用されているかどうか。

オプションを選択すると、その効果は[**OK**]で確認する前に効果を確認することのできるダイアログの下のプレビューペインに表示されます。

JSON 設定

選択する各オプションのために、効果はすぐダイアログの下のプレビューペインに反映されます。

(C) 2015-2021 Altova GmbH

- *インライン配列:* 配列のアイテムを単一のライン（またはインライン）に表示します。それ以外の場合、アイテムは個別のラインに表示されます。配列にインライン書式設定を決して適用しない、またはインライン書式設定を空の配列または特定のサイズまでの配列に適用することを選択することができます。
- *インラインオブジェクト:* オブジェクトのプロパティを単一のライン（またはインライン）に表示します。それ以外の場合、プロパティは個別のラインに表示されます。インライン書式設定をオブジェクトに決して適用しない、または空のオブジェクト、または、プロパティの特定の数を持つオブジェクトにインライン書式設定を適用することを選択できます。
- *インラインパディング:* 選択されると空以外のインライン配列と空以外のインラインオブジェクト間に空白を追加します。このオプションは空以外のインライン配列または空以外のインラインオブジェクトオプションが選択されている場合有効化されます。
- *インラインパディングを空にする:* 選択されると空のインライン配列の区切り文字の中に空白を追加します。配列またはオブジェクトが（空または空ではない）インラインに設定されている場合このオプションは有効化されます。
- *単一引用符に囲まれた文字列（JSON5）:* 選択されるとJSON5ドキュメント内のすべての引用符を一重引用符に変換します。
- *引用符で囲まれていない文字列（JSON5）:* 選択されている場合JSON5ドキュメント内で（`key:value`ペアの）全てのキーの周りから引用符を削除します。

上のチェックボックスを介してテキストビュー設定 969 ダイアログ内でインデントが使用されるかまたはインデントを使用しないかを指定することができます。

テキストビューの設定

テキストビュー をクリックしてインデント、ブックマーク余白、自動ハイライトなどのテキストビューのプロパティを有効化できるテキストビュー設定ダイアログを開きます。テキストビュー設定ダイアログはメニューコマンド [表示 | テキストビュー設定] 969 を介してアクセスすることができます。ダイアログについて説明されています。

24.14.12.6 ビュー

ビューセクションでは、XMLSpyにおけるXMLドキュメントの表示方法をカスタマイズすることができます。

プログラムロゴ

アプリケーションの起動時に表示されるスプラッシュ画面を無効にすることで、起動時間を短縮することができます。更に、ライセンスを購入されている場合、プログラムロゴ、著作権情報、登録に関する詳細の印刷をXMLSpyにて行わないように指定することができます。

ウィンドウタイトル

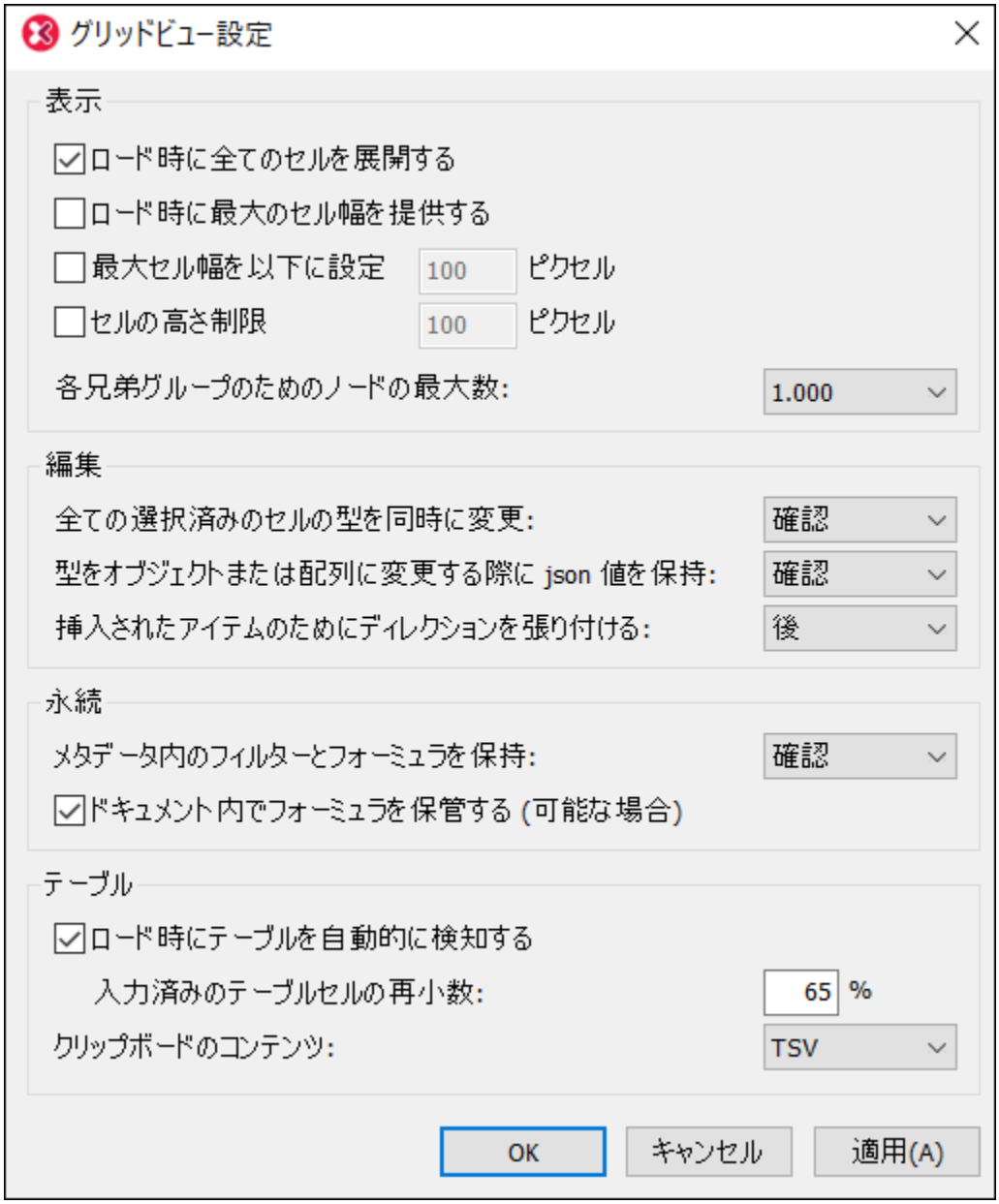
各ドキュメントウィンドウのためにウィンドウタイトルは、ファイル名、または、フルパス名になります。

テキストビューの設定

テキストビュー をクリックしてインデント、ブックマーク余白、自動ハイライトなどのテキストビューのプロパティを有効化できるテキストビュー設定ダイアログを開きます。テキストビュー設定ダイアログはメニューコマンド [表示 | テキストビュー設定] 969 を介してアクセスすることができます。ダイアログについて説明されています。

グリッドビュー設定

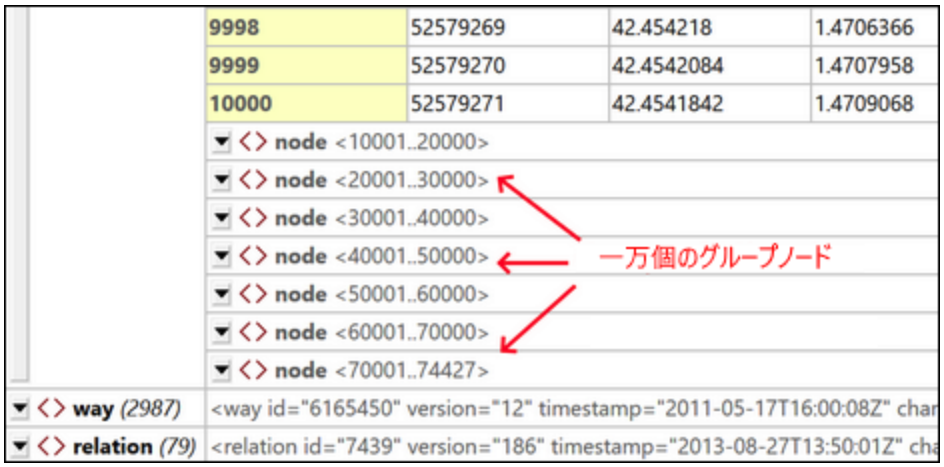
グリッドビュー設定 をクリックしてグリッドビュー設定ダイアログを開きます。グリッドビューのために以下のオプションをセットすることができます:



表示
表示 セクション内のチェックボックスは自明です。以下は更に説明が必要な場合のメモです。

- ロードする際に全てのセルが展開されていない場合ルートノードとそのすべての子孫は折りたたまれています。ドキュメントを更に深く移動するためには各ノードを展開する必要があります。
- 最適な幅がオンに切り替えられている場合グリッドの幅全体が表示されます。これを達成するには、セルの一部のテキストがラップされます。
- 兄弟ノードは100、1000 または10000 ノードの兄弟グループで表示することができます（*下のスクリーンショットを参照してください*）。多数の兄弟ノードを持つファイル内でスペースの表示を省略する場合

(C) 2015-2021 Altova GmbH



編集

編集 セクション内のチェックボックスは自明です。以下は更に説明が必要な場合のメモです。

- 複数の選択されたセルの型を変更する場合、設定のアクションを継続するかを決定する以下のオプションが与えられます: (ユーザーにより決定される) *常に*、*なし* または *確認* 。
- JSON 型を動的な型からオブジェクトまたは列挙に変更する場合、設定のアクションを継続するかを決定するかを決定する以下のオプションが与えられます: (i) *確認* (動的な型の値が名前を持たない子 `key:value` ペアの値として保持される、または破棄されるかを確認) (ii) *常に* (名前を持たない子 `key:value` ペア内の値を保持する) (iii) *なし* (値を保持する。すなわち値を破棄する)。
- *ディレクションの貼り付け* オプションはクリップボード内の選択範囲が選択されたセルの上または下に張り付けられるかを決定します。

永続

フィルターとフォーミュラを XML と JSON ドキュメントのメタデータ内に保管することができます。フォーミュラは XML ドキュメント内の処理命令として、そして JSON ドキュメント内のコメントとして保管されます。これらの設定はフィルターとフォーミュラの保管の方法を指定します。フォーミュラの場合 *可能な場合* という用語はフォーミュラが実際にすべての JSON ドキュメント (JSON5 と JSONC ドキュメント内でのみ) のコメントとして保存できないことを意味します 。

テーブル

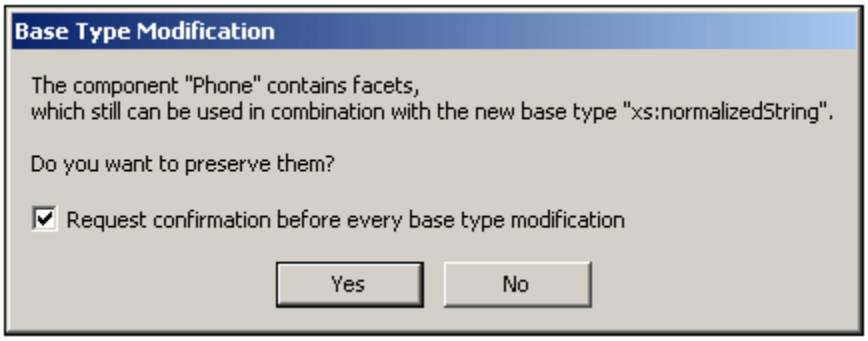
以下の設定を使用することができます:

- 設定がロード時にテーブルを自動的に検知する場合、テーブルをテーブルとして検知するために評価する記入されたテーブルセルの最小のパーセントを選択することができます。記入されたテーブルセルの数値がこのレベルを越さない場合、構造はそれぞれが下に表示されている繰り返し要素を持つ通常のグリッドとして表示されます。
- クリップボードコンテンツがタブにより区切られた値 (TSV) または (ドキュメントの型に従い) XML/JSON として保管するかを選択することが可能です。これはとても役に立つ機能です: クリップボードからテーブルを他のドキュメントに張り付ける場合、この設定により TSV としてまたはマークアップとしてテーブルがコピーされるかを選択できるようになります。(違いを確認するには、テーブルをそれぞれのフォーマットでクリップボードにコピーしテキストエディターにテーブルを張り付けてください。)

スキーマビュー

XML スキーマデータ型は他のデータ型から派生することもあります。例えば `xs:string` を基底型として、電子メールに使用される要素のデータ型を派生させることができます (`xs:string` データ型を特定の文字セットに制限します)。その後基底型が変更された場合、以下にあるオプションのいずれかを選択することができます:

- *コンテンツの保持*:新たな基底型でも既存の定義を使用することができる場合、このオプションを有効にすることで、定義が自動的に保持されます。.
- *変更を確認する*:基底型を変更すると、(*下のスクリーンショットに示される*)ダイアログが表示され、それまでの定義を保持したまま新たな基底型を使用し続けるか尋ねられます。



Authentic View

このオプションが有効になっている場合、**StyleVision Power Stylesheet** をベースにしたXML ファイルがAuthentic View にて自動的に開かれます。

ブラウザービュー

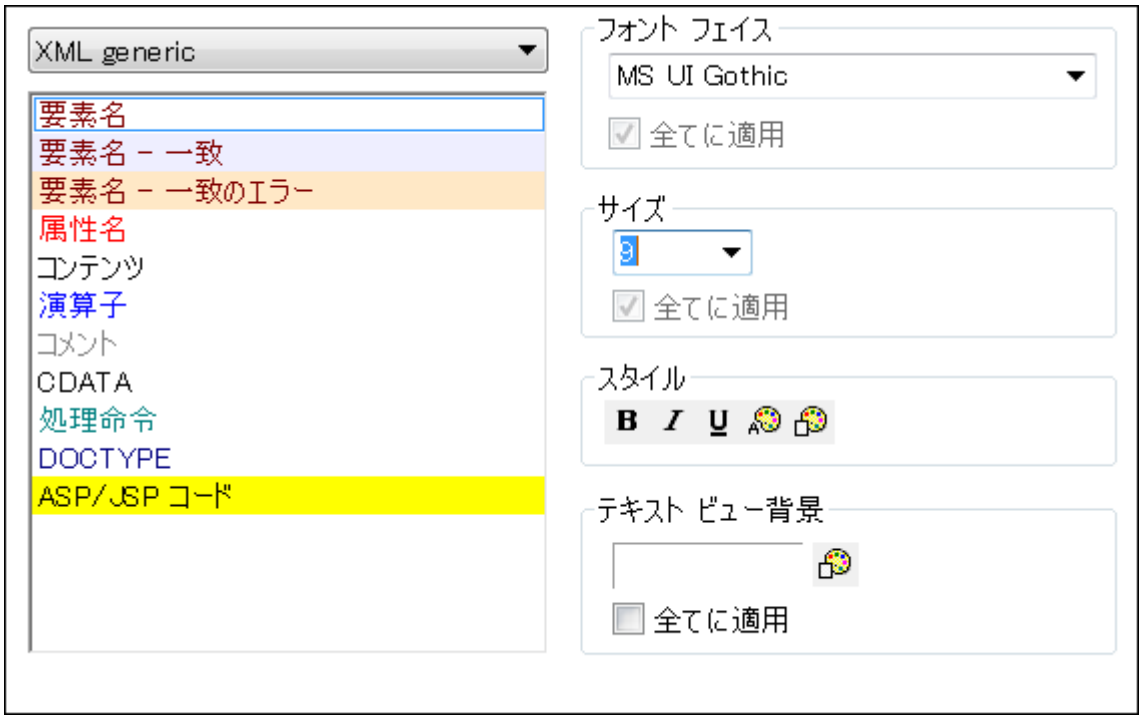
ブラウザービューを独立したウィンドウとして表示して、編集を行なっているビューウィンドウに隣合わせで表示することができます。

保存して完了

設定が完了したら、「OK」をクリックして完了します。

24.14.12.7 フォントと色

フォントと色 セクションでは、XMLSpy の異なるビュー内のテキストアイテムの外観のためのカスタム化オプションが与えられています。



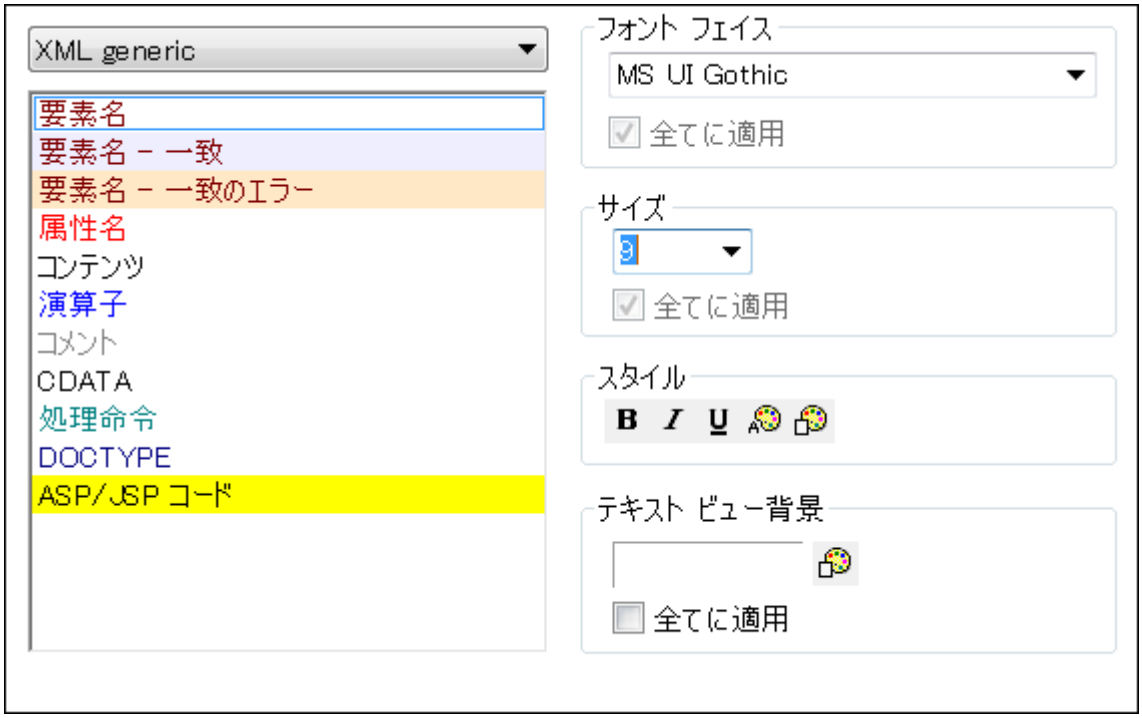
次のビューのためのオプションを使用することができます:

- テキストビュー 1029
- グリッドビュー 1031
- スキーマ設計ビュー 1034

ダイアログの左側のペイン内で、カスタマイズするビューを選択します。書式設定することのできるテキストアイテム型は、右側のペイン内に表示されます(*上のスクリーンショット*)。書式設定するテキストアイテム型を選択し、希望する書式設定のプロパティの値を割り当てます。

24.14.12.7.1 テキストビュー

例えば、要素名と属性名を別々の色で表示することができます。(*下のスクリーンショット および ドキュメントの型内のドキュメントのリストセクション内のドキュメント を参照してください*)。例えば、異なる書式設定を XML ドキュメント内の要素名と属性名の表示のために、または、XQuery ドキュメント内のキーワードと変数を表示するために指定することができます。



ドキュメントの型

次のドキュメントのテキストビューアイテムの書式設定を行うことができます:

- XML ジェネリック
- XQuery
- CSS
- JSON
- C-family
- Python
- XULE
- その他
- 出力

カスタム化の方法

以下の操作により、テキストフォントのカスタマイズを行うことができます:

1. 左上にあるコンボボックスにてカスタマイズしたいテキストフォントを選択します。この際、選択された種類のドキュメントで表示されるテキストアイテムがコンボボックス以下のボックスに表示されます(上のスクリーンショットでは、`XML generic` がドキュメントの種類として選択されています)。
2. カスタマイズを行うテキストアイテムクリックにより選択します。(*上のスクリーンショットでは* 要素名 *が選択されています。*)
3. 右側のペイン内のオプションを使用し、選択されたテキストアイテム型のフォントプロパティを設定します。*テキストビュー背景* オプションにより、テキストビュー全体のための背景色の選択が有効化されます。

以下の点に注意してください:

- (XML ジェネリックドキュメントの型などの) 特定のドキュメントの型のすべてのテキスト アイテム型のための同じフォントフェイス、サイズ、スタイルです。ドキュメントの型内で個別のテキスト型の、テキストの色と背景色のみを変更することができます。これにより構文色分け機能が有効化されます。
- *要素名* テキスト型は3つのサブ型により構成されています: (i) *要素名* は、選択されていない要素名に適用されます。(ii) *要素名 – 一致* は開始タグ名が終了タグ名が一致する箇所以外で選択されている要素名(カーソルが置かれている名前)に適用されます。(iii) *要素名 – 一致 エラー*は、選択された要素名に適用することができます。編集される要素名は、このために開始タグ名が終了タグ名 に一致するかに従い、異なる背景色にハイライトされます。景色に対応する背景色を変更することによりこれらのハイライトの色を変更することができます。デフォルトで、ハイライトはオンに切り替えられ、テキストビュー設定[969] ダイアログ内の*ハイライト 要素* オプションの選択を解除することによりオフに切り替えることができます。
- その他のカテゴリ: (i) アクティブな選択は、現在選択されているテキストのコンテンツを指します。アクティブでない選択は、同じテキストコンテンツのドキュメント内の他の箇所での発生を指します。(ii) アクティブなマーカーの検索は、現在選択されている検索の結果の発生を*指します*。マーカーの検索は、他の(アクティブではない) 検索結果の発生を指します。(iii) *デバッグ呼び出し マーカー*は、デバッグセッション内の現在選択されているステップを指します。(iv) *ハ表示されている空白* は、ドキュメント内の空白を指します。ドキュメント内の空白は空白マーカーの切り替え[969]により表示を切り替えることができます。

デフォルトに設定

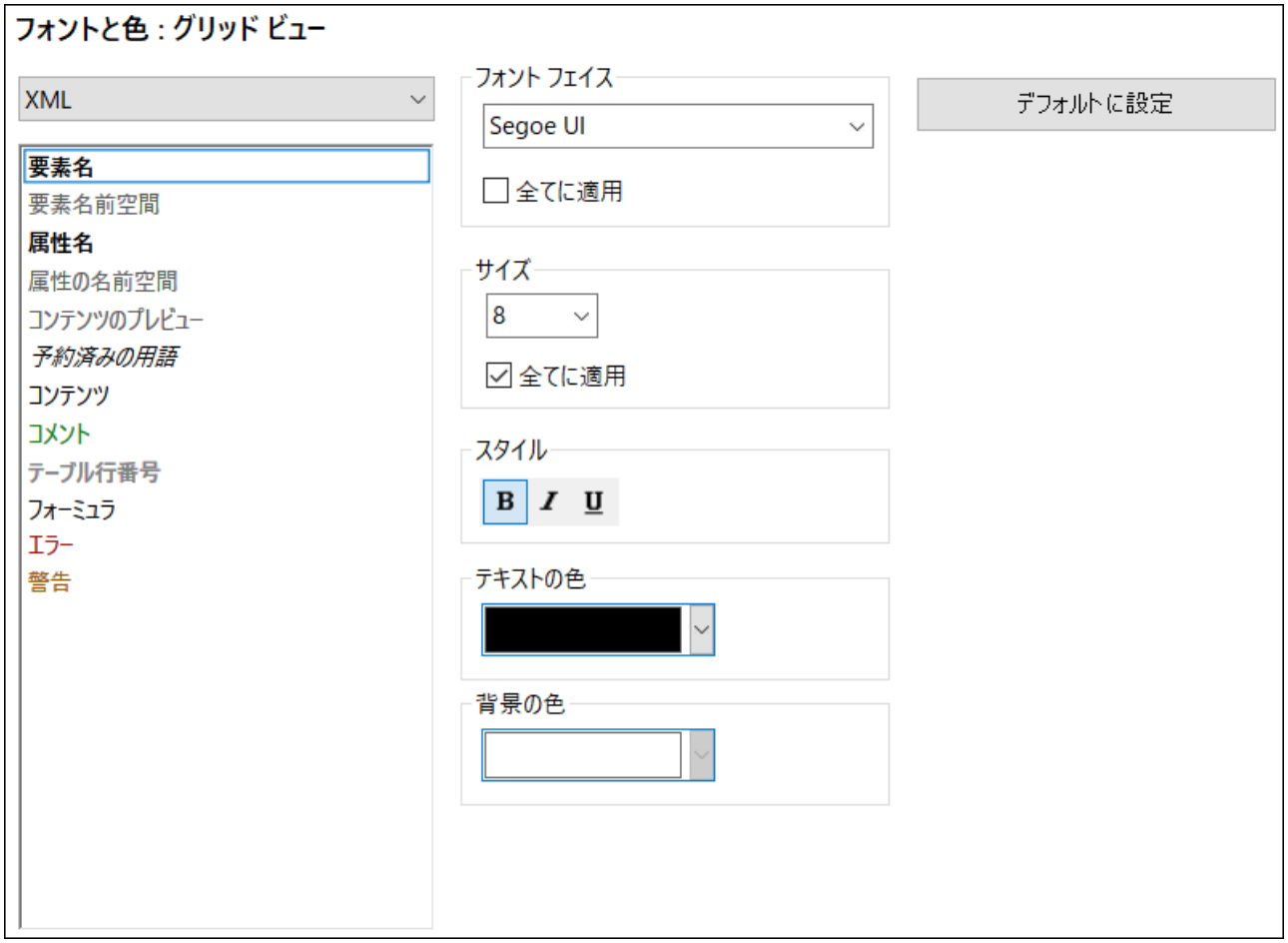
デフォルトに設定ボタンによりオリジナルのインストール設定にフォントがリセットされます。

保存して完了する

設定が完了したら、「OK」をクリックして完了します。

24.14.12.7.2 グリッドビュー

グリッドビューのセクション (*下のスクリーンショット*) によりXML グリッドビュー[141] とJSON グリッドビュー[475] 内のテキストの外観をカスタム化することができます。グリッドコンポーネントがテーブルで表示されている場合 *セルの色* オプション (*スクリーンショット*) によりセルの色を設定することができます。コンボボックス内でグリッドビューを構成するためのドキュメントの表示を選択します。書式設定するテキストの型を選択し、(*下にリストされる*) 希望するフォーマットプロパティに割り当てます。



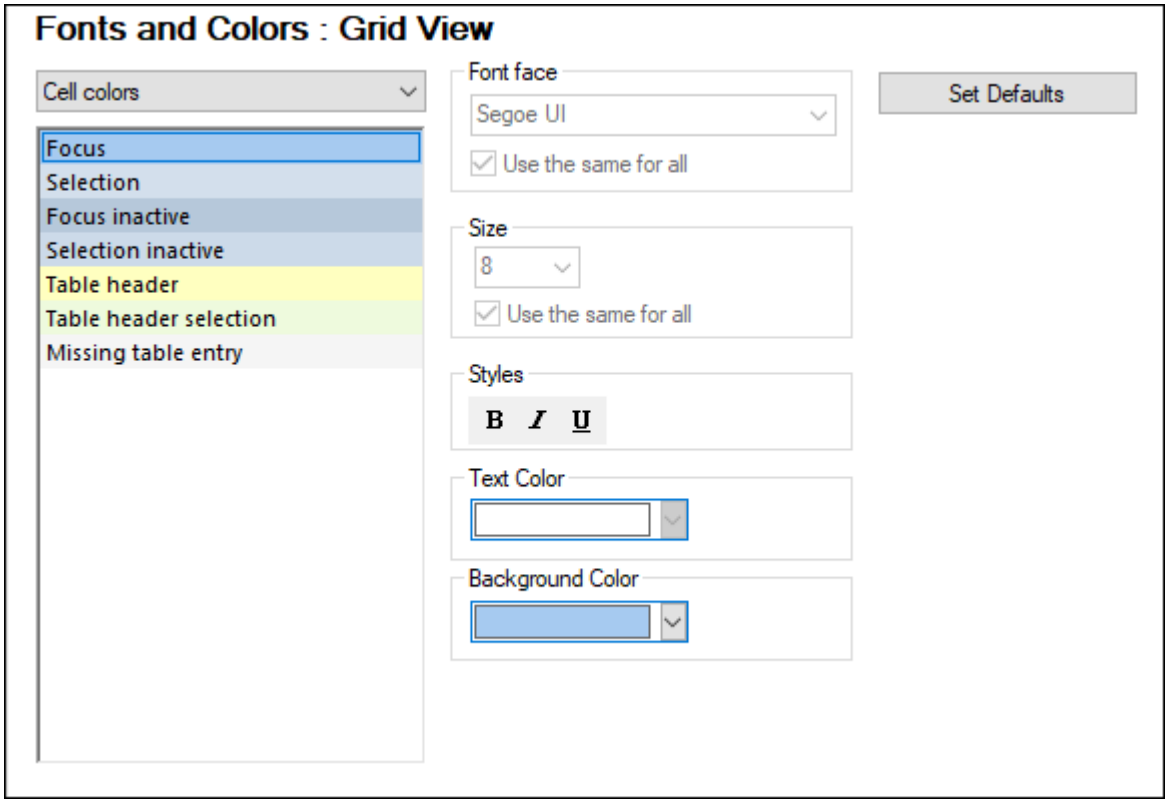
XML グリッド、JSON グリッド

選択されたテキスト アイテム型のために次のプロパティを設定することができます:

- *フォント フェイスとサイズ*: 選択されたフォントはグリッド ビューの印刷内でも使用することもできます。全てのテキスト アイテム型のために同じフォント フェイスまたはサイズを使用する場合、対応する、*「全てに対して同じ設定を使用する」* チェックボックスをチェックします。
- *テキスト スタイル、テキストの色、テキストの背景色*: 個々のテキスト アイテム型のスタイル、色、及び背景を設定します。現在の設定は左側のペインのリスト内で直ぐに反映され、テキスト アイテムをプレビューすることができます。

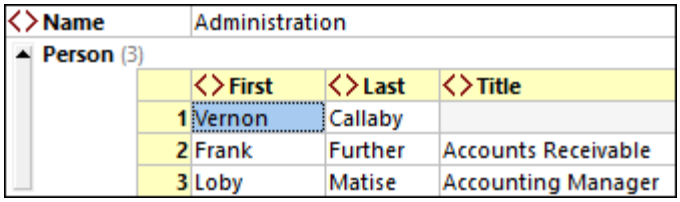
(全てのグリッド内のテーブルの表示の)セルの色

全てのグリッド (XML、JSON) 内のテーブル表示を言語に関連するセマンティックだけではなく設定することが可能です(例えば、XML 要素を XML コメントとは異なるように書式設定することが可能になります)。セル関数をベースにグリッド ビューをフォーマットすることもできます: 例えば、テーブル コンポーネントの選択された/選択を解除された状態の区別することなどができます。このような区別は異なる背景を各アイテムに割り当てることで達成することができますが(*下のスクリーンショット参照*)セルの異なる型のために他の/追加フォーマット プロパティを使用することもできます。

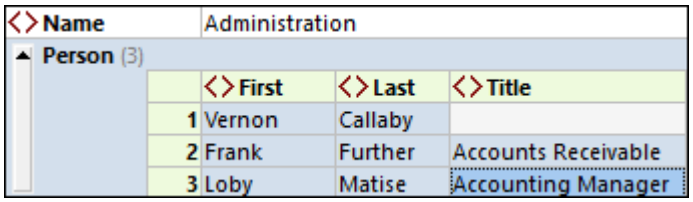


以下のコンポーネントのための設定を使用することができます:

- *(選択されていない)テーブルヘッダー* と *テーブルヘッダーの選択*: これらは列と行ヘッダーを参照します。下のスクリーンショットは選択を解除されたヘッダーを表示しています。背景色は上のダイアログで設定されている通りです。個別のヘッダーが選択されておらず、全てのヘッダーが選択されていると*選択された*ヘッダーの色が有効化されます。列と行が交わるセルをクリックすることですべてのヘッダーを選択、または、テーブルまたはその祖先のいずれかを選択することにより全てのヘッダーを選択することができます。



- *不足しているテーブルエントリ*: ドキュメント内の既存ではない要素または属性を参照します (*下のスクリーンショット参照*)。
- *選択* と *フォーカス*: 選択されたセルとフォーカスを持つセルを参照します。例えば、下のスクリーンショットでは `Person` テーブルの全体が選択されており、右下のセルにフォーカスが存在します。



- *非アクティブな選択、非アクティブなフォーカス*: セルのブロックが選択されており、そのブロック内の1つまたは複数のセルがフォーカスに設定されている場合、グリッドビューがアクティブな間は両方の選択範囲とフォーカスがアクティブです。しかしながら、グリッドビュー内の選択範囲またはフォーカスが削除されず他のダイアログまたはウィンドウがアクティブに設定された場合、グリッドビュー内の選択範囲とフォーカスは非アクティブになります。グリッドビューがアクティブに戻されると、選択範囲とフォーカスもまたアクティブに戻ります。

メモ　ここで定義された色以外にも、XMLSpy ではWindows の設定にて選択されたメニューの色も反映されます。

デフォルトに設定

デフォルトに設定ボタンによりオリジナルのインストール設定にフォントがリセットされます。

保存して完了する

設定が完了したら、「OK」をクリックして完了します。

24.14.12.7.3　スキーマ設計ビュー

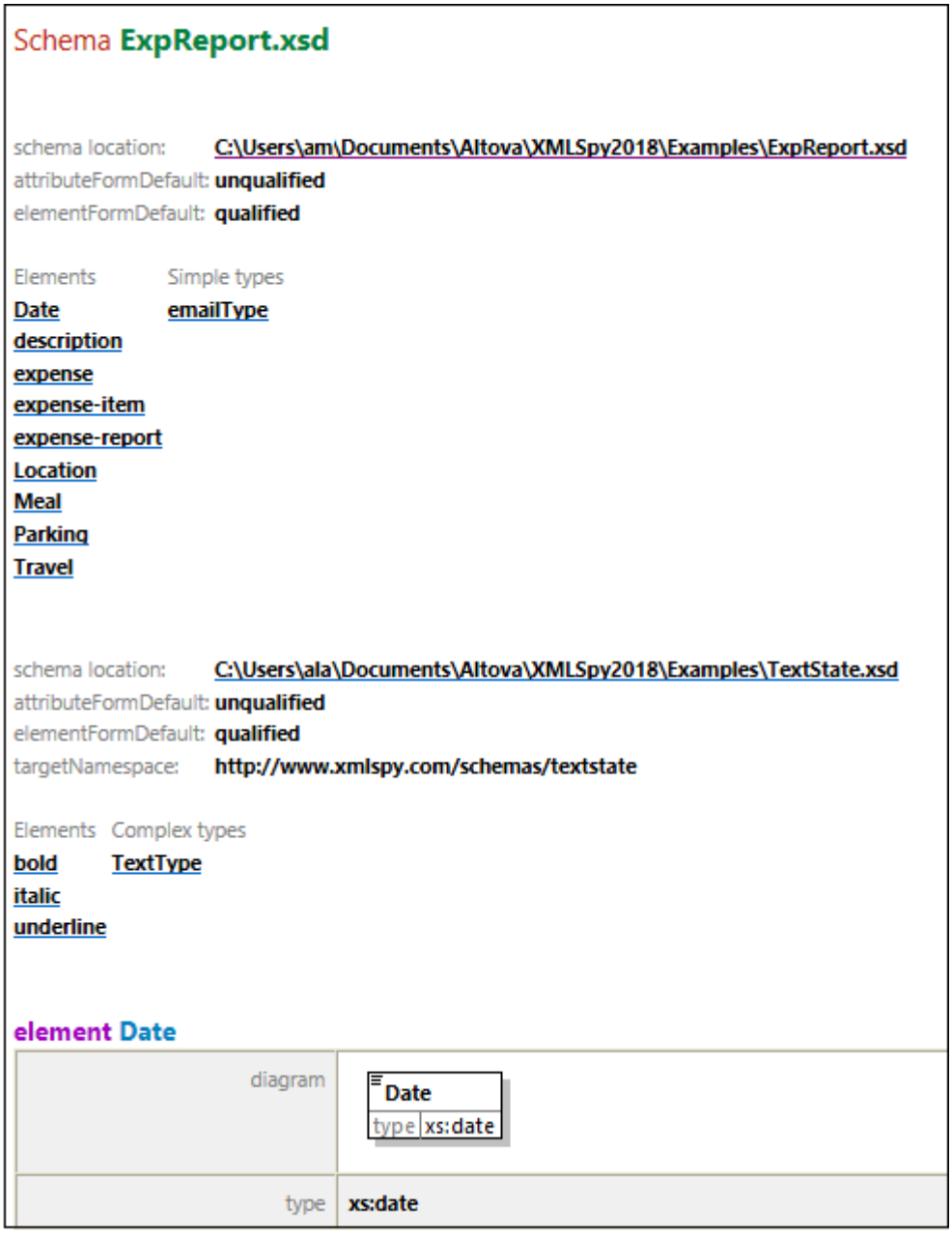
スキーマ設計ビュー セクションでXML スキーマ284 およびJSON スキーマ467 ドキュメントスキーマビュー151 の表示の外観をカスタム化することができます。

カスタム化の方法

選択されたドキュメントの型の個別のテキストアイテム型をカスタム化するには、次を行います:

1. 左上ののコンボボックスから、XML スキーマまたはJSON スキーマを選択します。選択されたドキュメントのテキストアイテム型がコンボボックスの下に表示されます。
2. 書式設定するテキストアイテム型をクリックして選択します。
3. 選択されたテキストアイテム型のプロパティを、右側のペイン内のオプションを使用して設定します。

メモ
ドキュメントスキーマヘッダー(2) 及び、ドキュメント要素ヘッダー(2) テキストアイテム型 は、それぞれ、スキーマの生成されたドキュメント 868 内のスキーマヘッダー、要素ヘッダーを指します。下のスクリーンショット内の生成されたドキュメントのスキーマヘッダーと要素ヘッダーの色を上のダイアログ内のプロパティの色と比較してください。



デフォルトに設定

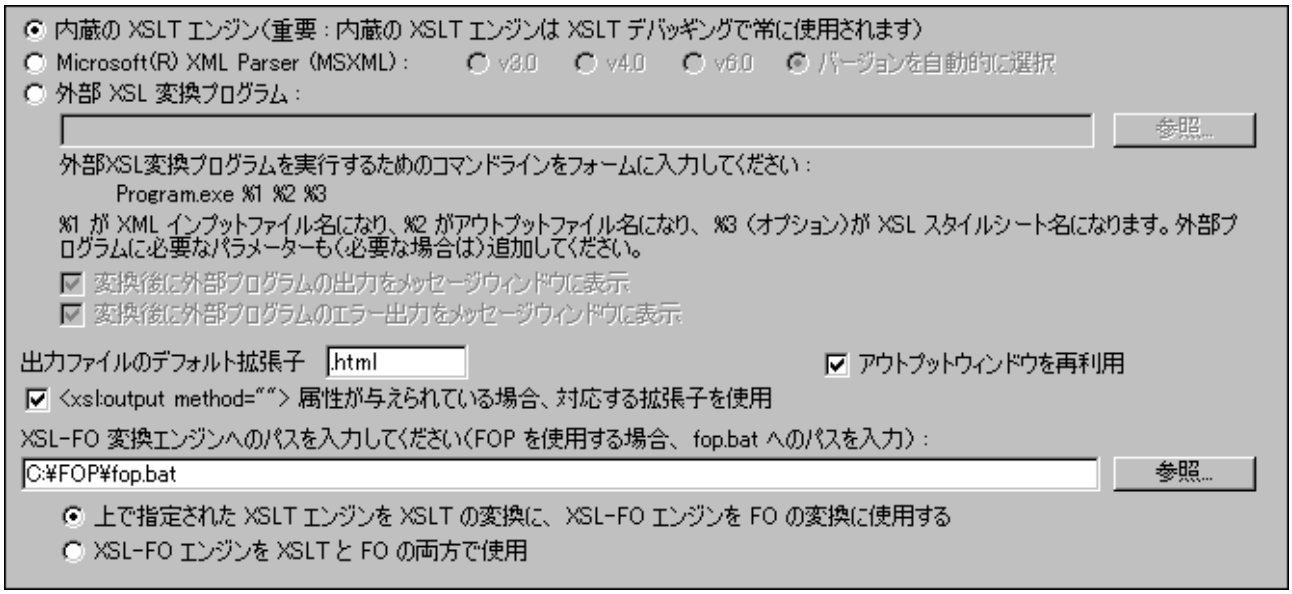
デフォルトに設定ボタンによりオリジナルのインストール設定にフォントがリセットされます。

保存して完了する

設定が完了したら、「OK」をクリックして完了します。

24.14.12.8 XSL

XSL セクションでは、アプリケーション内から行われる XSLT 変換[1036] と XSL-FO 変換[1037] におけるオプションを定義することができます



XSLT 変換

XMLSpy にはAltova XSLT 1.0 エンジンならびにAltova XSLT 2.0 エンジンが搭載されており、XSLT 変換に使用することができます。xsl:stylesheet または xsl:transform 要素の version 属性の値により、適切な XSLT エンジン(1.0 または 2.0)が変換には使用されます。これはXSLT 変換だけではなく、XMLSpy を使ったXSLT/XQuery デバッガーのデバッグでも適用されます。

XSLT を使ったXML ドキュメントの変換には、以下のどれかを使用することができます:

- 内蔵の Altova XSLT エンジン(Altova (XSLT 1.0、XSLT 2.0、およびXSLT 3.0)。
- MSXML 3.0、4.0、または 6.0 パーサー：お使いのコンピューターで使用されている MSXML のバージョンが明らかな場合、選択を行うことができ、そうでない場合、アプリケーションによるバージョンの選択を行わせるべきでしょう(バージョンを自動的に選択オプションがデフォルトでは選択されています)。このオプションが選択されている場合、利用できる最新バージョンが選択されます。
- 選択した外部 XSLT プロセッサー：外部 XSLT プロセッサーを指定するコマンドライン文字列を入力する必要があります。コマンドライン文字列を構築する際に、以下の変数を使用することができます：

  %1 = 処理する XML ドキュメント
  %2 = 生成する出力ファイル

%3 = 使用する XSLT スタイルシート（XML ドキュメント内に、スタイルシートへの参照が含まれない場合）

例えば以下のコマンドにより、Saxon (XSLT 1.0) を使った単純な変換を行うことができます：

```
myxsltengine.exe -o output.xml input.xml stylesheet.xslt parameter-name=parameter-value
```

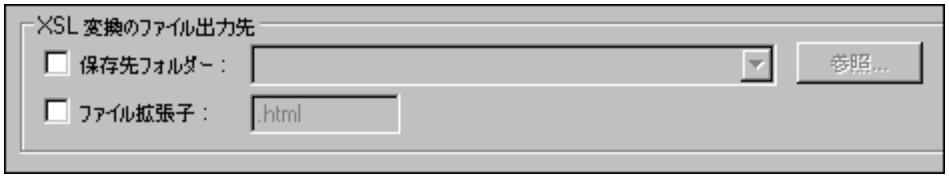
XMLSpy からこのコマンドを実行するには、外部 XSL 変換プログラムラジオボタンを選択し、以下の文字列をテキストボックスに入力してください。

```
c:\myxsltengine\myxsltengine.exe -o %2 %1 %3 parameter-name=parameter-value
```

適切なチェックボックスにチェックを入れることで、外部プログラムから得られた出力やエラーメッセージを XMLSpy のメッセージウィンドウに表示することができます。.

メモ: XMLSpy のXSLT 入力パラメーターダイアログ 390 にてセットされたパラメーターは、内蔵のAltova XSLT エンジンに対してのみ渡されます。デフォルトとしてセットされたその他のXSLT エンジンに値は渡されません。

アウトプットウィンドウを再利用オプションを使用することで、後に続く変換の結果を、同じ出力ウィンドウにて表示することができます。XML ファイルがプロジェクトに格納されており、アウトプットウィンドウを再利用オプションが無効になっている場合、この設定はプロジェクトプロパティ 808 の出力ファイルパスにおける保存先フォルダーも無効になっている場合のみ適用されます。



XSL-FO 変換

FO ドキュメントはFO プロセッサーにより処理され、FO プロセッサーの実行可能ファイルへのパスは、XSL-FO 変換エンジンのテキストボックスにて指定する必要があります。変換処理は、「 XSL/XQuery | XSL-FO 変換 888 」メニューコマンドから行うことができます。ソースファイル(IDE 内部からコマンドが指定された場合は、アクティブなドキュメント）がXSL-FO ドキュメントの場合、変換にFO プロセッサーが呼び出されます。ソースドキュメントがXML ドキュメントの場合、まずはXSLT 変換を使ってXML ドキュメントをXSL-FO ドキュメントへ変換する必要があります。このXSLT 変換では、アプリケーションのデフォルトエンジンとして指定したXSLT エンジンのほか、デフォルト FO プロセッサーとして指定したFO プロセッサー内部に組み込まれたXSLT エンジンを使用することもできます(上を参照 1036)。これらオプションを選択するには、適切なラジオボタンをクリックしてください。

設定が完了したら、「OK」をクリックして、オプションダイアログを閉じてください。

メモ: Apache XML プロジェクト のFOP プロセッサをインストールするオプションを解除しない限り、次のフォルダー内にインストールされます C:\ProgramData\Altova\SharedBetweenVersions。インストールされると、XSL-FO エンジン入力ボックス内にパスが自動的に入力されます。FO プロセッサへのパスを希望するように設定することができます。しかしながら、FO プロセッサを使用する他のAltova 製品により同じパスがにより使用される場合、FO プロセッサを選択して設定をすることもできます（StyleVision とAuthentic Desktop）。

保存して完了する

設定が完了したら、「OK」をクリックして、オプションダイアログを閉じてください。

24.14.12.9 XQuery

XQuery タブは、XQuery とXQuery Update ドキュメントの編集と実行に関連するオプションを定義します（*下のスクリーンショット*）。

一般的な XQuery オプション

以下のオプションから選択することができます:

- *シリアル化:* シリアル化は、XQuery ドキュメントが実行された時に生成される出力ドキュメントを指します。シリアル化 メソッド（アダプティブ*、HTML、JSON、テキスト、XHTML、またはXML）とシリアル化エンコード、および、入力の異なる型のシリアル化の暗号化を選択することができます。出力のシリアル化は、XML 入力とJSON 入力のために個別に選択することができます（*メモ* アダプティブなメソッドにより、エラー無しにインスタンスドキュメントを処理することができます。入力ドキュメントをベースにしてシリアル化メソッドを自動的に決定します）。
- *XML 宣言の証言:* シリアル化された（出力）ドキュメント内のXML 宣言を省略します。
- *インデント 出力:* 出力ドキュメントをインデントして、ドキュメント階層構造を表示します。
- *XML ソースを常にスキップする:* XQuery ドキュメントが実行されると、XMLSpy は、XQuery ドキュメントを実行するためのXML ソースを問うことができます。プロンプトダイアログによりXML ファイルを参照することができます。このオプションをスキップして、XQuery ドキュメントを直接実行することもできます。このオプションが選択されると、XQuery ドキュメントは、XML ドキュメントにパスされることなく正確に実行されます。これはXML ドキュメントが必要とされないため、またはXML データがXQuery ドキュメント内の関数によりアクセスされるためです。
- *XML ファイルの検証:*XQuery ドキュメントの実行内で使用されるXML ファイルを検証します。無効な XML ファイルがフラッグされ、XQuery ドキュメントが処理されません。
- *DB2 行の取得 :* DB データを表示で、取得される行の最大数を指定することができます。XMLSpy は、`.xqr` ファイル拡張子[1018] をXQuery-for-DB ファイルとして認識します。

- *XQuery デフォルトのバージョン:* XQuery ドキュメントの実行のために使用する `version` キーワードを持たない XQuery エンジンバージョンを指定します。これは XQuery と XQuery Update ドキュメントの両方に適用され、使用するデフォルトの XQUery エンジンを選択します。

XQuery Update オプション

次の XQuery Update オプションを使用することができます:

- *更新:* XQuery Update ファイルが実行されると、ターゲット XML ファイルは直接ディスク上で更新することができます。または、XMLSpy 内で開き、メモリ内で更新することもできます。更新時にファイルを開くオプションにより、ファイルを更新し、ディスクにファイルを保存することができます。また、(保存せずに) 更新を拒否することもできます。
- *元のフォーマットを保持する:* 更新されたドキュメントの元のフォーマットを可能な限り保管します。

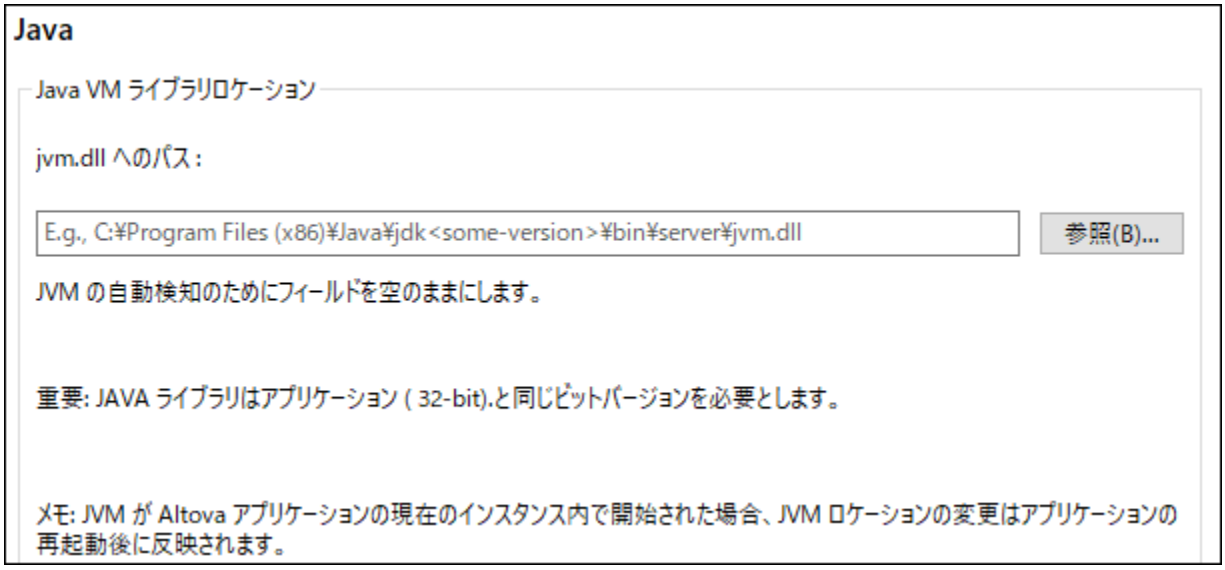
保存して完了します。

設定が完了したら、「OK」をクリックして完了します。

24.14.12.10 Java

Java タブ上では、ファイルシステム上で Java VM (仮想マシン) へのパスを任意で入力することができます。カスタム Java VM パスの追加は常に必要なわけではありません。デフォルトで、XMLSpy は、Windows レジストリと JAVA_HOME 環境変数を (この順序で) 読み取ることにより、Java VM パスを自動的に影響しようとします。このダイアログボックスに追加されるカスタムパスは、自動的に検知される VM パスより先優先順位を与えられます。

カスタム Java VM パスを追加する必要がある場合、例えば、インストーラを持たず、(例えば、Oracle の OpenJDK などの) レジストリエントリを作成する必要のない Java 仮想マシンを使用している場合、カスタム Java VM パスを追加する必要がある場合があります。XMLSpy により自動的に検知された Java VM パスを何らかの理由でオーバーライドする場合、このパスを設定する必要があります。



以下の点に注意してください:

- Java VM パスは Altova デスクトップと (サーバーではない) アプリケーションにより共有されています。この結果、アプリケーション内

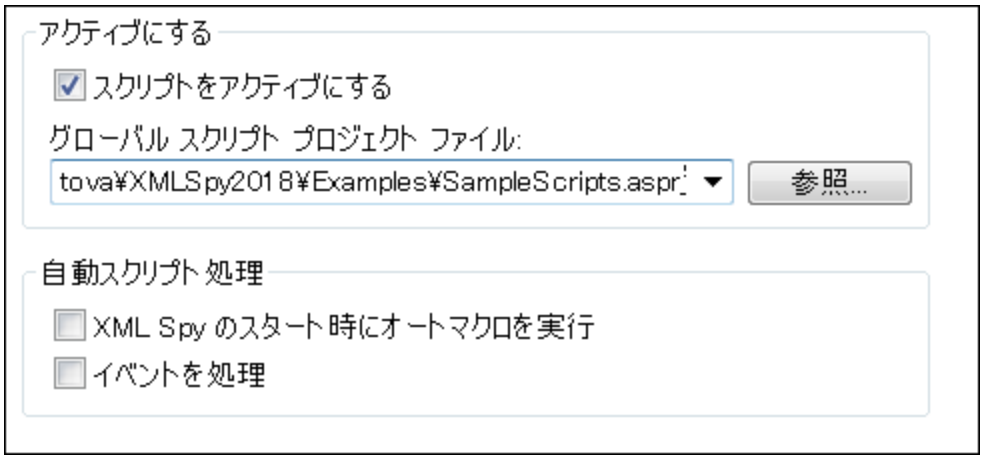
で変更されると、他のAltova アプリケーションに自動的に適用されます。
- パスはJDK がインストールされているディレクトリに相対する**\bin\server** または**\bin\client** ディレクトリからの**jvm.dll** ファイルを指す必要があります。
- XMLSpy プラットフォーム(32-ビット、64-ビット) がJDK と同じである必要があります。
- Java VM パスの変更後、新規の設定を反映するためにXMLSpy を再起動する必要があります。

Java VM パスの変更は以下のエリアに影響を与えます:

- JDBC 接続性
- XSLT/XPath のためのJava 拡張関数

24.14.12.11 スクリプト

スクリプトタブでは、アプリケーションの起動時にスクリプト環境[974] を有効にすることができます。有効にするには、*スクリプトをアクティブにする* チェックボックスにチェックを入れてください(以下のスクリーンショットを参照)。



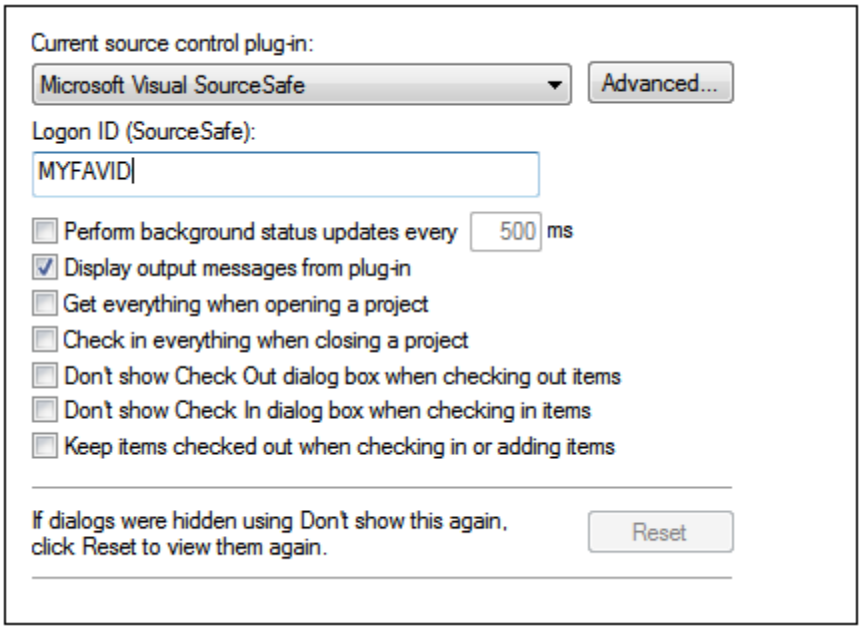
XMLSpy に対してグローバルスクリプトプロジェクトを設定するには、スクリプトをアクティブにする チェックボックスにチェックを入れた後、必要な Altova スクリプトプロジェクト (.asprj) ファイルを指定します。その他にも、(i) XMLSpy の起動時にマクロを自動的に実行するか、(ii) プロジェクト内のアプリケーションイベントハンドラーを自動的に実行するかを有効に(または無効に)することができます。

保存して完了する

「OK」をクリックして完了してください。グローバルリソーススクリプトプロジェクト内のマクロがマクロコマンドのサブメニューに表示されます。

24.14.12.12 ソース管理

ソース管理セクションではソース管理プロバイダーの指定を行い、デフォルトのログイン ID やユーザー設定を各種ソース管理プロバイダーに対して指定することができます。



ソース管理プラグイン

現在インストールされている中から、使用するソース管理プラグインをコンボボックスのドロップダウンリストを使って選択することができます。目的のソース管理を選択したら、次のテキストボックスにてログイン ID を入力します。「**詳細**」ボタンをクリックすることで、選択されたソース管理プラグインに特化したダイアログが表示され、ソース管理プラグインに対して設定を行うことができます。これら設定の内容はソース管理プラグインにより変化します。

ユーザー設定

以下のようなユーザー設定を利用することができます:

- ユーザーが指定した時間間隔ごとに、バックグラウンドでステータスの更新を行うか、この機能を無効にすることもできます。巨大なソース管理データベースを使用する場合、多くの CPU パワーならびにネットワークリソースが消費されることになります。バックグラウンドのステータス更新機能を無効にするか、インターバルの時間を長く取ることで、システムのスピードを図ることができます。
- プロジェクトを開くならびに閉じる時に、それぞれファイルのチェックアウトやチェックインを自動的に行うことができます。
- チェックアウトならびにチェックインを行う際に、ダイアログの表示を無効にすることができます。
- ダイアログボックスにて次回から表示しないオプションを選択した場合、リセットボタンが有効になります。リセットボタンをクリックすることで、ダイアログが再度表示されるようになります。

保存して完了

設定が完了したら、「OK」をクリックして完了します。

24.14.12.13 ネットワーク プロキシ

ネットワーク プロキシ セクションでは、カスタムのプロキシの設定を構成することができます（*下のスクリーンショット*）。デフォルトは、システムのプロキシの設定が使用されますが、設定はユーザーの介入無しで作動します。代替のネットワークプロキシを設定する場合、設定を定義するオプションを使用してください。

システムプロキシの設定の使用

システムプロキシ設定を介して構成可能な インターネット エクスプローラー（IE） 設定を使用します。`netsh.exe winhttp` を介して構成される設定が必要とされます。

自動プロキシの構成

以下のオプションを使用することができます:

- *自動検知の設定:* DHCP または DNS を使用して W PAD スクリプト（`http://wpad.LOCALDOMAIN/wpad.dat`）を検索し、プロキシセットアップのためにこのスクリプトを使用します。
- *スクリプト URL:* プロキシセットアップのために使用されるプロキシ自動構成（`.pac`）スクリプトに対する HTTP URL を指定します。
- *再ロード:* 現在の自動プロキシ構成をリセットして再ロードします。このアクションには W indows 8 または以降が必要とされ、30 秒程の時間が必要です。

手動のプロキシの構成

ホスト名とポートを対応する製品のプロキシのために手動で指定します。サポートされるスキームはホスト名に含まれている場合があります（例: `http://hostname`）。プロキシがスキームをサポートする場合、対応するプロトコールと同じである必要はありません。

以下のオプションを使用することができます:

(C) 2015-2021 Altova GmbH

- *HTTP プロキシ:* HTTP プロトコルのための指定済みのホスト名とポートを使用します。*全てのプロトコルのためにこのプロキシサーバーを使用する* が選択されている場合、HTTP プロキシがすべてのプロトコルのために使用されます。
- *SSL プロキシ:* SSL プロトコルのために指定されているホスト名とポートが使用されます。
- *プロキシ無し:* セミコロン（`;`）により区別されているプロキシを使用しないホスト名、ドメイン名、または、ホストのための IP アドレスのリスト。IP アドレスは切り捨てられず、IPv6 アドレスは角かっこで囲まれる必要があります（例: `[2606:2800:220:1:248:1893:25c8:1946]`）。ドメイン名は、ドットと共に開始される必要があります（例: `.example.com`）.
- *プロキシのサーバーをローカルのアドレスのために使用しない*：チェックされている場合、*プロキシ無し* リストのために `<local>` を追加します。このオプションが選択されている場合、次の場合、プロキシは使用されません（i）`127.0.0.1`、(ii) [::1]、(iii) (`.`) ドット文字を含んでいないホスト名すべて。

現在のプロキシの設定

プロキシの検知の詳細なログを提供します。*URL のテスト* フィールドの右の「**更新**」ボタンを使用して更新することができます（例、URL のテスト を変更する場合、または、プロキシの設定が変更された場合）。

- *URL のテスト:* URL のテストを使用して、どのプロキシが特定の URL ために使用されているかを確認することができます。URL を使用して、I/O はされません。プロキシの自動構成が選択されている場合、このフィールドは空にしておく必要があります（*システムプロキシの設定の使用*、または、*自動プロキシの構成* を使用して）。

24.15 ウィンドウメニュー

ウィンドウメニューには、殆どのWindows アプリケーションに備わっているコマンドが含まれており、XMLSpy セッションで開かれた個々のドキュメントウィンドウを管理することができます。

開かれているドキュメントウィンドウを重ねて表示したり、上下や左右に並べて表示、最小化したドキュメントウィンドウへアクセスすることができます。各種入力ヘルパーを有効/無効にすることができるほか、ウィンドウダイアログから、目的のドキュメントウィンドウへ直接アクセスすることもできます。

24.15.1 重ねて表示

このコマンドにより開かれている全てのドキュメントウィンドウが重ね合わさって表示されるように配置されます。

24.15.2 上下に並べて表示

このコマンドにより、開かれている全てのドキュメントが上下に並べて配置され、複数のドキュメントを同時に閲覧することができます。

24.15.3 左右に並べて表示

このコマンドにより、開かれている全てのドキュメントが左右に並べて配置され、複数のドキュメントを同時に閲覧することができます。

24.15.4 プロジェクトウィンドウ

このコマンドによりプロジェクトウィンドウ 105 の表示/非表示を切り替えることができます。

プロジェクトウィンドウはドッキングすることのできるウィンドウで、タイトルバーをドラッグすることで、オリジナルの位置から切り離してフロートウィンドウにすることができます。タイトルバーを右クリックしてウィンドウをドッキングしたり隠したりすることができます。

24.15.5 情報ウィンドウ

このコマンドにより情報ウィンドウ[107] の表示/非表示を切り替えることができます。

情報ウィンドウはドッキングすることのできるウィンドウで、タイトルバーをドラッグすることで、オリジナルの位置から切り離してフロートウィンドウにすることができます。タイトルバーを右クリックしてウィンドウをドッキングしたり隠したりすることができます。

24.15.6 入力ヘルパー

このコマンドにより全ての入力ヘルパーの表示/非表示を切り替えることができます。

3つある入力ヘルパーウィンドウはドッキングすることのできるウィンドウで、タイトルバーをドラッグすることで、オリジナルの位置から切り離してフロートウィンドウにすることができます。タイトルバーを右クリックしてウィンドウをドッキングしたり隠したりすることができます。

24.15.7 出力ウィンドウ

出力ウィンドウは、(検証結果のようなメッセージを表示する)メッセージウィンドウや、(XPath 条件式評価の結果を表示する)XPath ウィンドウをタブ化したウィンドウです。初期状態ではメインウィンドウの下部に表示されます。「**出力ウィンドウ**」コマンドにより、出力ウィンドウの表示/非表示を切り替えることができます。

出力ウィンドウのウィンドウはドッキングすることのできるウィンドウで、タイトルバーをドラッグすることで、オリジナルの位置から切り離してフロートウィンドウにすることができます。タイトルバーを右クリックしてウィンドウをドッキングしたり隠したりすることができます。

出力ウィンドウに関する詳細は、テキストビューセクションの出力ウィンドウ[103] を参照ください。

24.15.8 プロジェクトおよび入力ヘルパー

このコマンドによりプロジェクトウィンドウならびに入力ヘルパーの表示/非表示を選択することができます。

24.15.9 全てオン/オフ

このコマンドにより、以下にあるドッキング可能な全てのウィンドウの表示/非表示を切り替えることができます:

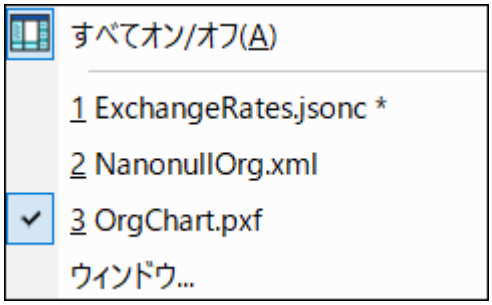
- プロジェクトウィンドウ[105]
- 情報ウィンドウ[107]

- 3つの入力ヘルパーウィンドウ
- 出力ウィンドウ 103

このコマンドによりドキュメント以外の全ウィンドウを素早く隠して、ドキュメントの表示を作業エリア全体に広げることができます。

24.15.10 現在開かれているウィンドウのリスト

このリストは現在開かれているすべてのウィンドウを表示し、ウィンドウ間で素早く切り替えることができます。



また**CTRL+F6** キーボードショートカットを使用して、開かれているウィンドウ間を循環することができます。

(C) 2015-2021 Altova GmbH

24.16 ヘルプメニュー

ヘルプメニューにはヘルプならびにXMLSpy に関する更に詳しい情報を得るために必要なコマンドが、ウェブサーバー上で利用できる情報へのリンクや、サポートページへのリンクとともに収められています。

ヘルプメニューからは登録ダイアログ[1049]を起動することもでき、購入した製品のライセンスキーを入力することができます。

24.16.1 目次、インデックス、検索

- 目次
  - *説明*

    ヘルプウィンドウの左側のペインに目次を表示した、XMLSpy の画面上のヘルプマニュアルを開きます。目次はヘルプドキュメント全体の概要を表示しています。目次のエントリをクリックしてトピックに移動することができます。

- インデックス
  - *説明*

    ヘルプウィンドウの左側のペインにキーワード インデックスを表示したXMLSpy の画面上のヘルプマニュアルを開きます。目次はヘルプドキュメント全体の概要を表示しています。インデックスはキーワードをリストし、キーワードをダブルクリックすることでトピックへ移動することができます。キーワードが 1 つ以上のトピックにリンクされている場合は、トピックのリストが表示されます。

- 検索
  - *説明*

ヘルプウィンドウの左側のペインに検索ダイアログを表示したXMLSpy の画面上のヘルプマニュアルを開きます。単語を検索するには、入力フィールドに検索対象を入力して、(i)「Return」を押す、または(ii)「**トピックのリスト**」をクリックします。を押します。ヘルプシステムは、ヘルプドキュメント全体で全文検索を行いヒットしたリストを返します。アイテムを表示するためにはアイテムをダブルクリックします。

24.16.2　キーボードマップ

「**ヘルプ | キーボードマップ**」コマンドにより、XMLSpy 内にある全てのコマンドが、メニューごとに分けられて表示されます。各メニューコマンドには、その説明とキーボードショートカットが表示されます。

コマンド	キー	説明
ファイル(F)エンコード(E)...		現在のファイルの文字セットエンコードを設定または変更する
ファイル(F)メールで送信(M)...		Eメールでドキュメントを送信します
ファイル(F)上書き保存(S)	Ctrl+S	アクティブなドキュメントを上書き保存します
ファイル(F)全て保存(L)		開いている全てのドキュメントを保存します
ファイル(F)全て閉じる		開いている全てのドキュメントを閉じます
ファイル(F)再ロード(D)		開いているファイルを再ロードします
ファイル(F)印刷(P)...	Ctrl+P	アクティブなドキュメントを印刷します
ファイル(F)印刷プレビュー(V)		印刷プレビュー
ファイル(F)印刷設定(R)...		プリンターと印刷の設定を変更します
ファイル(F)名前を付けて保存(...		アクティブドキュメントを名前を付けて保存します
ファイル(F)新規作成(N)...	Ctrl+N	新しいドキュメントを作成します
ファイル(F)終了(X)		アプリケーションを終了します：ドキュメントを保存するか問い合...
ファイル(F)閉じる(C)		アクティブなドキュメントを閉じます
ファイル(F)開く(O)...	Ctrl+O	既存のドキュメントを開きます
ファイル(F)非アクティブを全て...		アクティブなドキュメントを除く全てのドキュメントを閉じる

特定のメニューに収められているコマンドを確認するには、カテゴリーコンボボックスからメニュー名を選択します。印刷アイコンをクリックすることで、コマンドを印刷することができます。

ショートカットに関するメモ:

- いくつかのコマンド（及びそのキーボードショートカット）は、特定のビューでしか有効になりません。例えば、XMLメニューにあるほとんどのコマンドは、グリッドビューでしか使用できません。その他のコマンド（例：「**ファイル | 保存**」や「**XML | 整形式のチェック**」）は複数のビューで使用することができます。
- 例えば、テキストビューやスキーマビューで「Shift + F10」を押下することでコンテキストメニューを表示したり、テキストビューにてカーソルが要素の開始または終了タグにあるときに「Ctrl + E」をう押下することで、対となるタグへ移動することができます。
- カスタマイズダイアログの キーボードタブ[1003] では、各種メニューコマンドに対して独自のショートカットをセットすることができます。

24.16.3 ライセンス登録、注文フォーム、登録、最新情報のチェック

▼ ソフトウェアのライセンス認証

説明

Altova 製品ソフトウェアをダウンロードすると、無料評価キーまたは購入されたライセンスキーを使用して、製品にライセンスを供与、または、ライセンスの認証を行うことができます。

- *無料 評価ライセンス*初めて製品のダウンロードとインストールを行うと、ソフトウェアライセンス認証ダイアログが表示されます。ダイアログでは無料 評価 ライセンスをリクエストすることができます。ユーザーの名前、所属会社名、そして電子メールアドレスを表示されるダイアログに入力し **リクエスト** をクリックします。ライセンスファイルが入力された電子メールアドレスに送信されます。この手順には数分を要します。ライセンスファイルを適切な場所に保存します。**リクエスト** をクリックすると、リクエストダイアログの下に入力フィールドが表示されます。このフィールドはライセンスファイルの パスを取ります。ライセンスファイルを参照 またはライセンスファイルへの パスを入力し「OK」をクリックします。(「ソフトウェアのライセンス認証ダイアログ」内で「**新規のライセンスをアップロードする**」をクリックしてライセンスファイルへの パスを入力するダイアログにアクセスすることができます。)ソフトウェアは30日の間アンロックされます。
- *永続的なライセンスキー*:ソフトウェアライセンス認証ダイアログには永続的なライセンスキーを購入するためのボタンが含まれています。このボタンをクリックすると、製品の永続的なライセンスキーを購入することのできる Altova オンラインショップに移動することができます。受信する電子メールはライセンスデータを含むライセンスファイルの書式で送信されています。3つの種類の永続的なライセンスが存在します:インストール済み、同時使用ユーザー、名前を持つユーザー。*インストール済みのライセンス* は単一のコンピューター上のソフトウェアのロックを解除します。*インストール済みのライセンス を*N 台のコンピューターのために購入するとN 台までのコンピューター上でソフトウェアを使用することができます。*同時使用ユーザーライセンス*はN 人の同時使用ユーザーにN 人のユーザーに同時にソフトウェアを使用することを許可します 。(10N 台のコンピューターにソフトウェアをインストールすることができます)。 *名前を持つユーザーライセンス*は*特定のユーザー*が5台の異なるコンピューター上でソフトウェアを使用することを許可します。ソフトウェアのライセンスを認証するには「**新規のライセンスをアップロードする**」をクリックして表示されるダイアログ内でライセンスファイルを参照、またはライセンスファイルの パスを入力し「OK」をクリックします。

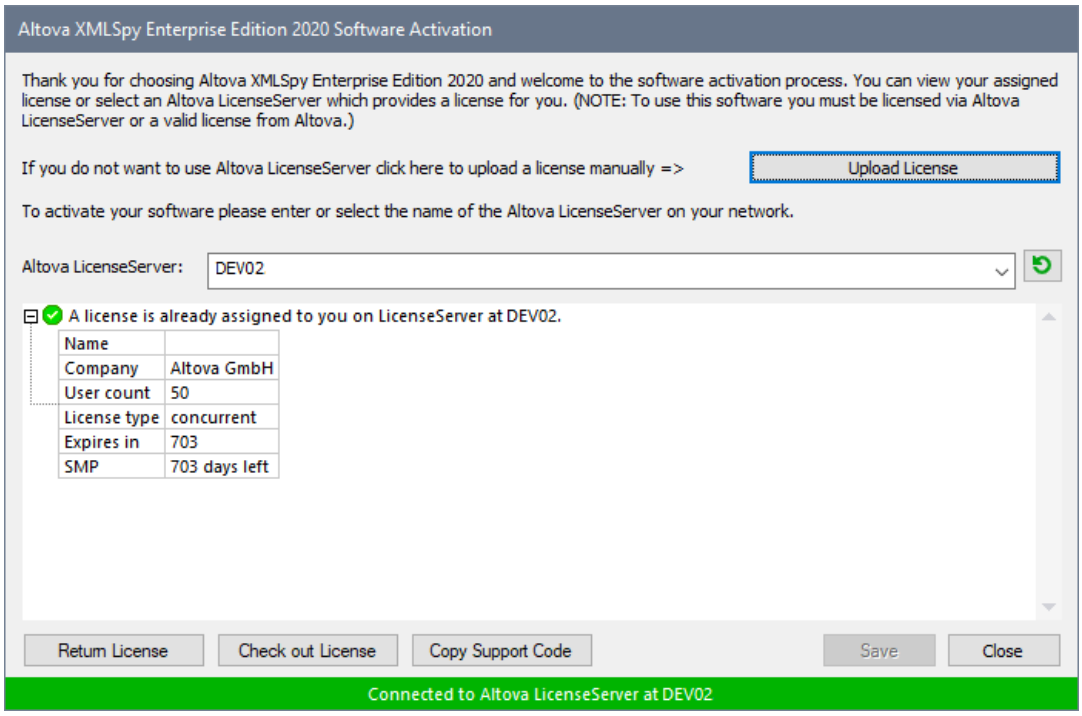
メモ 複数のユーザーライセンスのために各ユーザーは各自の名前を入力するようにプロンプトされます。

> *ライセンスの電子メールと Altova 製品へのライセンス供与(有効化) の異なる方法*
> Altova から受信するライセンス電子メールにはライセンスファイルが添付ファイルとして含まれています。ライセンスファイルは `.altova_licenses` ファイル拡張子を有しています。
>
> Altova 製品のライセンスを認証するには、以下のうち1つ行ってください:
>
> - 適切な場所にライセンスファイル(`.altova_licenses`)を保存し、ライセンスファイルをダブルクリックし、表示されるダイアログに必要な情報を入力し、「**キーの適用**」をクリックして完了します。
> - ライセンスファイル(`.altova_licenses`)を適切な場所に保存します。Altova 製品内では、「**ヘルプ | ライセンス登録**メニューコマンドを選択し、**新規のライセンスをアップロード**します。ライセンスファイルへの パスを入力し「OK」をクリックします。
> - 適切な場所にライセンスファイル(`.altova_licenses`)を保存し、Altova LicenseServer のライセンスプールにアップロードします。以下を行うことができます:(i) 製品のソフトウェアライセンス認証ダイアログから Altova 製品からライセンスを取得します。*(以下を参照)*または(ii) Altova LicenseServer から製品へのライセンスを割り当てます。*LicenseServer の使用の詳細に関しては、下記のトピックを参照してください。*

ソフトウェアライセンス認証ダイアログ(*下のスクリーンショット*) は、「**ヘルプ | ソフトウェア アクティベーション**」をクリックすることにより常にアクセスすることができます。

以下の方法によりソフトウェアをアクティブ化することができます:

- *ソフトウェアライセンス認証ダイアログでライセンスを登録する方法。*ダイアログ内で、「**新規のライセンスをアップロード**」をクリックして、ライセンスファイルを参照し選択します。「OK」をクリックしてライセンスファイルへのパスを確認します(複数のユーザーライセンスの場合は個人の名前です)。「**保存**」をクリックして完了します。
- *ネットワーク上のAltova LicenseServerを使用してライセンス供与する方法:* ネットワーク上のAltova LicenseServerを使用してライセンスを取得するにはソフトウェアのライセンスの認証ダイアログの下にある**Altova LicenseServerを使用する**をクリックします。使用するLicenseServerがインストールされているマシンを選択します。License Serversの自動検知はLAN上で配信が送信されることを意味します。これらの配信がサブネットに制限されているため、License Serverは自動検知のためのクライアントマシンと同じサブネット上に存在する必要があります。自動検知が作動しない場合、サーバーの名前を入力します。Altova LicenseServerはライセンスプール内でAltova製品のためのライセンスを有している必要があります。LicenseServerプール内に存在する場合、ソフトウェアライセンス認証ダイアログ内に表示されます(*Altova XMLSpy内のダイアログで表示されている例を参照してください*)。「**保存**」をクリックしてライセンスを取得します。



マシン固有のライセンスがLicenseServerからインストールされると、7日間は、LicenseServerに戻すことができません。7日過ぎると、(「**ライセンスを戻す**」をクリックして)マシンのライセンスをLicenseServerに戻すことができ、このライセンスは、他のクライアントによりLicenseServerから取得することができます。LicenseServer管理者は、LicenseServerのWeb UIを使用して、取得されたライセンスの割り当てを解除することができます。ライセンスの返却は、マシン固有のライセンスのみに適用され、現在使用中のライセンスには適用されないことに注意してください。

ライセンスのチェックアウト

ライセンスが製品マシン上に保管されるように、ライセンスをライセンスプールから30日間チェックアウトすることができます。これにより、オフラインで作業することが可能になります。この機能はとても役に立ちます。Altova LicenseServerにアクセスできない環境(例えば、旅行中にAltova製品がインストールされたラップトップコンピューターで作業する)場合などが挙げられます。ライセンスはチェックアウトされていますが、LicenseServerは、ライセンスが使用中と表示し、ライセンスは他のマシンで使用することができません。ライセンスはチェックアウトの期間が終わると自動的にチェックインされた状態を戻します。または、チェックアウトされたライセンスはソフトウェアのライセンスの認証ダイア

ログのボタンを使用して「**チェックイン**」することができます。

ライセンスをチェックアウトするには以下をおこないます: (i) ソフトウェアのライセンスの認証ダイアログで「**ライセンスのチェックアウト**」をクリックします(*上のスクリーンショット参照*)。(ii) ライセンスのチェックアウトダイアログ内から、チェックアウトの期間を選択し、「**チェックアウト**」をクリックします。ライセンスがチェックアウトされます。ライセンスのチェックアウト後2つの状態が発生します: (i) ソフトウェアのライセンス認証ダイアログは時刻およびチェックアウトの期限を含むチェックアウトに関する情報を表示します。(ii) ダイアログ内の「**ライセンスのチェックアウト**」ボタンは「**チェックイン**」ボタンに変更されます。「**チェックイン**」ボタンをクリックして、ライセンスをチェックインすることができます。チェックアウト期間の期限が切れると、ライセンスは自動的にチェックイン状態に戻されるため、選択したチェックアウトの期間がオフラインで作業する期間をカバーするように確認してください。

メモ ライセンスのチェックアウトを可能にするには、LicenseServer 上でチェックアウト機能が有効化されている必要があります。チェックアウトを試みる際この機能が有効化されていない場合、エラーメッセージが表示されます。この場合、LicenseServer 管理者に連絡してください。

サポートコードのコピー
「**サポートコードのコピー**」をクリックして、ライセンスの詳細をクリップボードにコピーしてください。これはオンラインサポートフォームを使用してサポートをリクエストする際に必要なデータです。

Altova LicenseServer を使用することにより、IT 管理者は、リアルタイムでネットワーク上の全てのライセンスの概要、および、クライアントの割り当てと、クライアントのライセンスの使用状況を確認することができます。LicenseServer を使用する利点は、ですから、多数の Altova ライセンスを管理することのできる管理機能です。Altova LicenseServer は、Altova Web サイトで無料で提供されています。Altova LicenseServer および Altova LicenseServer を使用したライセンスの供与に関する詳細は、Altova LicenseServer ドキュメントを参照してください。

注文フォーム

説明

ソフトウェア製品のライセンス許与バージョンを注文する準備が整っている場合、(*前のセクション参照*) ソフトウェアライセンス認証ダイアログ内の「**永久ライセンスの購入**」ボタン、または「**注文フォーム**」コマンドを使用して Altova オンラインショップに移動して注文することができます。

登録

説明

Altova 製品登録ページをブラウザーのタブに表示します。Altova ソフトウェアを登録することにより、最新の製品の情報が得られます。

更新のチェック

説明

Altova サーバーに接続して、より新しいバージョンの製品が利用可能かどうかチェックし、その結果を表示します。

24.16.4 他のコマンド

サポートセンター

説明

インターネット上にあるAltova サポートセンターへのリンクとなっています。サポートセンターにはFAQ やディスカッションフォーラムが含まれており、問題の解決方法を探り、Altova の技術サポートスタッフへアクセスすることができます(現在英語のみの提供となります)。

WEB 上のFAQ

説明

インターネット上にあるAltova のFAQ へのリンクとなっています。FAQ データベースはAltova のサポートスタッフにより常時更新されています。

コンポーネントのダウンロード

説明

インターネット上にあるAltova のコンポーネントダウンロードセンターへのリンクとなっています。このリンク先から様々なコンポーネントソフトウェアをダウンロードして、Altova 製品とともに使用することができます。ソフトウェアコンポーネントはXSLT やXSL-FO プロセッサーからアプリケーションサービスプラットフォームまで、幅広く提供されています。コンポーネントダウンロードセンターにてご利用になれるソフトウェアは、通常無料でご利用になれます。

インターネット上のXMLSpy

説明

インターネット上にあるAltova Web サイトへのリンクとなっています。Altova Web サイトでは、XMLSpy や関連するテクノロジーについて確認することができます。

XMLSpy トレーニング

説明

Altova Web サイトでオンライントレーニングページを見つけることができますAltova の専門家トレーナーによるオンラインコースを選択することができます。

XMLSpy について

説明

スプラッシュ画面と製品のバージョン番号が表示されます。XMLSpy の64 ビットバージョンを使用している場合、これはアプリケーション名の後のサフィックス(x64) により示されています。32 ビットバージョンにはサフィックスは存在しません。

(C) 2015-2021 Altova GmbH

24.17 コマンドライン

コマンドラインからXMLSpy に搭載されているいくつかのコマンドを実行することができます。以下のコマンドを利用することができます:

ファイルを開く

コマンド: `xmlspy.exe file.xml`
動作: XMLSpy にてファイルfile.xml を開く。

メモ: XML ファイルに SPS が割り当てられている場合、XML ファイルは Authentic ビューにて開かれます。そうでない場合、XML ファイルはテキストビューにて開かれます。SPS ファイルが割り当てられていない場合、/sps フラグにより割り当てを行うことができます(下記参照)。

複数のファイルを開く

コマンド: `xmlspy.exe file1.XML file2.xml`
動作: XMLSpy にてファイルfile1.xml ならびにfile2.xml を開く。

SPS ファイルを XML ファイルに割り当てて、Authentic ビューによる編集を行う

コマンド: `xmlspy.exe myxml.xml /sps mysps.sps`
動作: mysps.sps をSPS ファイルとして、ファイルmyxml.xml をAuthentic ビューで開く。/sps フラグにより、あとに続く SPS ファイルが/sps フラグの前にある XML ファイルに(Authentic ビュー編集を行うために)割り当てられます。

SPS ファイルから新たな XML テンプレートを開く

コマンド: `xmlspy.exe mysps.sps`
動作: 新規 XML ファイルをAuthentic ビューで開きます。表示はSPS をベースにしたもので、SPS スキーマをベースにした新規 XML ファイルが作成されます。XML ファイルの保存を行う際に、ファイルの名前を指定する必要があります。

テキストビューで SPS ファイルを XML ドキュメントとして開く

コマンド: `xmlspy.exe /raw mysps.sps`
動作: テキストビューで、ファイルmysps.sps をXML ドキュメントとして開きます。/raw フラグにより、それに続く SPS ファイルがXML ファイルとして編集されます。

25 プログラマーのレファレンス

XMLSpy は自動化サーバーです: 自動化クライアントは直接にオブジェクトおよび自動化サーバーにより使用することのできる機能にアクセスすることができます。自動化クライアントは直接にオブジェクトおよび自動化サーバーにより使用することのできる機能にアクセスすることができます。ですから、XMLSpy の自動化クライアントは、例えば、XMLSpyのXML 検証機能を使用することができます。。この結果、開発者は、XMLSpy の既存の機能を使用してアプリケーションを拡張することができます。

XMLSpy, のプログラムすることのできるオブジェクトは、XMLSpy のアプリケーション API を使用して、自動化クライアントで使用することができます。XMLSpy のアプリケーション API は、以降省略してアプリケーション API と呼ばれます。アプリケーション API のオブジェクトモデル、および使用可能なすべてのオブジェクトの詳細はこのドキュメントで説明されています(Application API 1095 のセクションを参照してください)。

実行環境

アプリケーション API は、以下の環境の内部からアクセスすることができます:

- スクリプトエディター 1056
- IDE プラグイン 1081
- 外部プログラム 1095
- ActiveX 統合 1413

これらの環境について下で簡単に説明がされています。

スクリプトエディター: XMLSpy 機能のカスタム化と変更
XMLSpy のインストールを機能を変更および追加することでカスタム化することができます。ユーザー入力のためのフォームを作成し、新しいメニューコマンド およびツールバーショートカットを含むユーザーインターフェイスを変更することもできます。これらすべての機能は、アプリケーション APIのオブジェクトとの作動するスクリプトを書くことにより達成することができます。これらのタスクを効率的に実行するために、XMLSpy は、ビルドインのスクリプトエディターを搭載しています。機能の完全な説明と使用方法はこのドキュメントのスクリプトエディター 1056 セクションで説明されています。サポートされるプログラミング言語は、**JScript** および **VBScript**. です。

IDE プラグイン: XMLSpy のためのプラグインの作成
XMLSpy により自身のプラグインを作成し、XMLSpy に統合することができます。これを行うには、**XMLSpy** のプラグインのための特別なインターフェイスを使用します。プラグイン作成に関する詳細は、XMLSpy IDE プラグイン 1081 のセクションで説明されています。

アプリケーション オブジェクトは、IDE プラグインにより実装され、アプリケーションにより呼び出される必要があるメソッドにパスされます。IDE プラグインの実装のために使用される 典型的な言語は、C# および C++ です。詳細に関しては、XMLSpy IDE プラグイン 1081 のセクションを参照してください。

外部プログラム

さらに、外部スクリプトともに**XMLSpy** を操作することができます。例えば、XMLSpy を指定した時刻に起動し、**XMLSpy** 内のファイルを開き、ファイルを検証し、印刷するスクリプトを作成することができます。外部スクリプトは、アプリケーション API を使用して、これらのタスクを実行します。アプリケーション API の詳細に関しては、アプリケーション API 1095 のセクションを参照してください。

XMLSpy からApplication API を使用するには、XMLSpy のインスタンスが最初に開始される必要があります。そして、Authentic Desktop インスタンスと関連するApplication オブジェクトを返します。個別の言語に関する情報は、プログラミング 言語 1097 のセクションを参照してください。

(C) 2015-2021 Altova GmbH

基本的には、XMLSpy はCOM 登録を使用して起動されます。そして、XMLSpy インスタンスと関連する Application オブジェクトを返します。COM 設定により、既に実行中のXMLSpy に関連したオブジェクトが返されます。Access to COM オブジェクトへのアクセスは、自動的にラップされたC# を使用します。共通する項目は、下にリストされています。

- JScript[1118] およびVBScript[1106] スクリプト ファイルは、シンプルな構文を持ち、COM オブジェクトにアクセスするためにデザインされています。DOS コマンド ライン から直接、またはW indows Explorer をダブルクリックして実行することができます。これらのスクリプトはシンプルな 自動化 タスクのために使用されます。
- C#[1109] は、広範囲の既存の機能を搭載した完全なプログラミング言語です。Access to COM オブジェクトへのアクセスは、自動的にラップされたC# を使用します。
- C++ は、COM への直接的なコントロールを与えますが、他の言語よりも多量のコードが必要です。
- Java[1118]: Altova 製品には、アプリケーション API をラップして、Java の完全な外観と使用感を与える、ネイティブのJava クラスが搭載されています。
- 役に立つ代替の他のプログラミング 言語は、次のとおりです: アプリケーション、Perl、およびPython のためのVisual Basic

ActiveX 統合

XMLSpy ActiveX コントロールを介してApplication API にアクセスする特別なケースです。この機能はXMLSpy 統合パッケージ[1413] がインストールされている場合のみ使用することができます。すべてのActiveX コントロールには、元になる機能のための対応するCOM オブジェクトを返すプロパティがあります。マネージャーコントロールは Application オブジェクト、ドキュメントコントロール、Document オブジェクト、およびプレースホルダオブジェクトを与え、プロジェクトツリーを含む箇所では、 Project オブジェクトを返します。これらのオブジェクトによりサポートされるメソッドは、アプリケーション API のインターフェイスのセクション[1130]で説明されています。ActiveX コントロール統合 のコンテキストでは、このメソッドを使用する必要がなくこの点を考慮する必要があります。詳細に関してはActiveX 統合[1413]を参照してください。

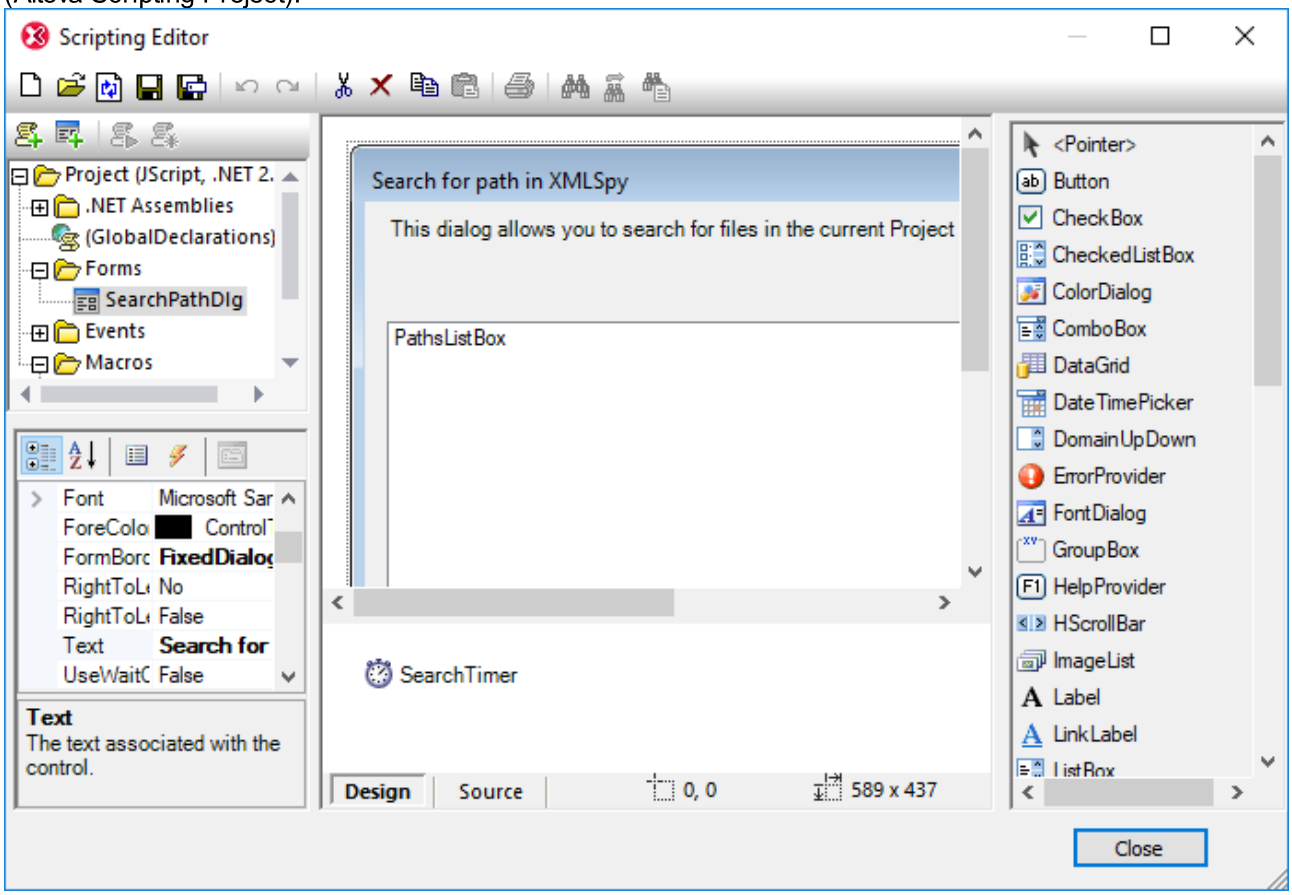
プログラマーのレファレンスについて

XMLSpy のプログラマーのレファレンスは、次のセクションから構成されています:

- スクリプトエディター[1056]:XMLSpy で使用することのできるスクリプト環境のためのユーザー参照。
- IDE プラグイン[1081]:XMLSpy のための プラグイン作成の説明
- Application API[1095]:アプリケーション API のためのレファレンス
- ActiveX 統合[1413]:ActiveX コントロールを使用した、XMLSpy GUI およびXMLSpy 機能の統合の方法の案内および参照。

25.1 Scripting Editor

Scripting Editor is a development environment built into XMLSpy from where you can customize the functionality of XMLSpy with the help of JScript or VBScript scripts. For example, you can add a new menu item to perform a custom project task, or you can have XMLSpy trigger some behavior each time when a document is opened or closed. To make this possible, you create scripting projects—files with .asprj extension (Altova Scripting Project).



Scripting Editor

Scripting projects typically include one or several macros—these are programs that perform miscellaneous custom tasks when invoked. You can run macros either explicitly from a menu item (or a toolbar button, if configured), or you can set up a macro to run automatically whenever XMLSpy starts. The scripting environment also integrates with the XMLSpy COM API. For example, your VBScript or JScript scripts can handle application or document events such as starting or shutting down XMLSpy, opening or closing a project, and so on. Scripting projects can include Windows Forms that you can design visually, in a way similar to Visual Studio. In addition, several built-in commands are available that help you instantiate and use .NET classes from VBScript or JScript code.

Once your scripting project is complete, you can enable it either globally in XMLSpy, or only for specific projects.

Scripting Editor requires .NET Framework 2.0 or later to be installed before XMLSpy is installed.

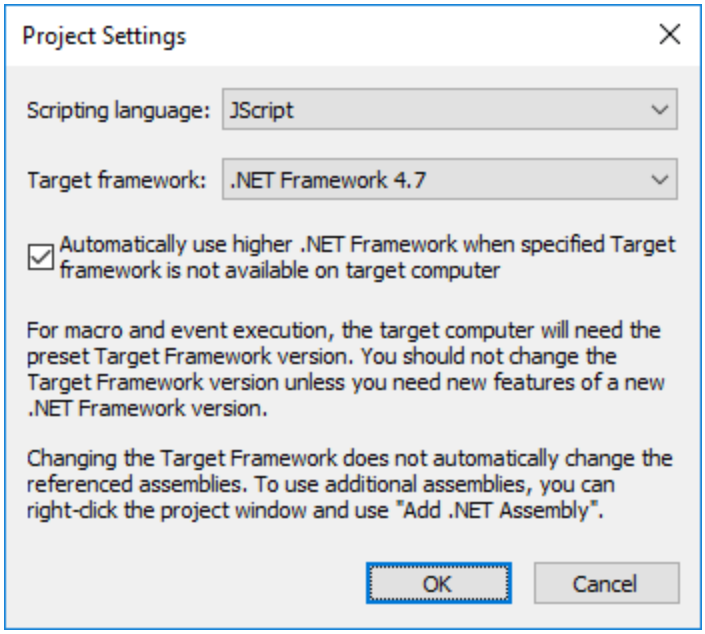
25.1.1 Creating a Scripting Project

All scripts and scripting information created in the Scripting Editor are stored in Altova Scripting Projects (.asprj files). A scripting project may contain macros, application event handlers, and forms (which can have their own event handlers). In addition, you can add global variables and functions to a "Global Declarations" script—this makes such variables and functions accessible across the entire project.

To start a new project, run the menu command **Tools | Scripting Editor**.

The languages supported for use in a scripting project are JScript and VBScript (not to be confused with Visual Basic, which is not supported). These scripting engines are available by default on Windows and have no special requirements to run. You can select a scripting language as follows:

1. Right-click the **Project** item in the upper-left pane, and select **Project settings** from the context menu.
2. Select a language (JScript or VBScript), and click **OK**.



From the Project settings dialog box above, you can also change the target .NET Framework version. This is typically necessary if your scripting project requires features available in a newer .NET Framework version. Note that any clients using your scripting project will need to have the same .NET Framework version installed (or a later compatible version).

By default, a scripting project references several .NET assemblies, like `System`, `System.Data`, `System.Windows.Forms`, and others. If necessary, you can import additional .NET assemblies, including assemblies from .NET Global Assembly Cache (GAC) or custom .dll files. You can import assemblies as follows:

1. Statically, by adding them manually to the project. Right-click **Project** in the top-left pane, and select **Add .NET Assembly** from the context menu.
2. Dynamically, at runtime, by calling the `CLR.LoadAssembly` 1072 command from the code.

You can create multiple scripting projects if necessary. You can save a scripting project to the disk, and then load it back into the Scripting Editor later. To do this, use the standard Windows buttons available in the toolbar: **New**, **Open**, **Save**, **Save As**. Once the scripting project has been tested and is ready for deployment,

you can load it into XMLSpy and run any of its macros or event handlers. For more information, see Enabling Scripts and Macros 1079.

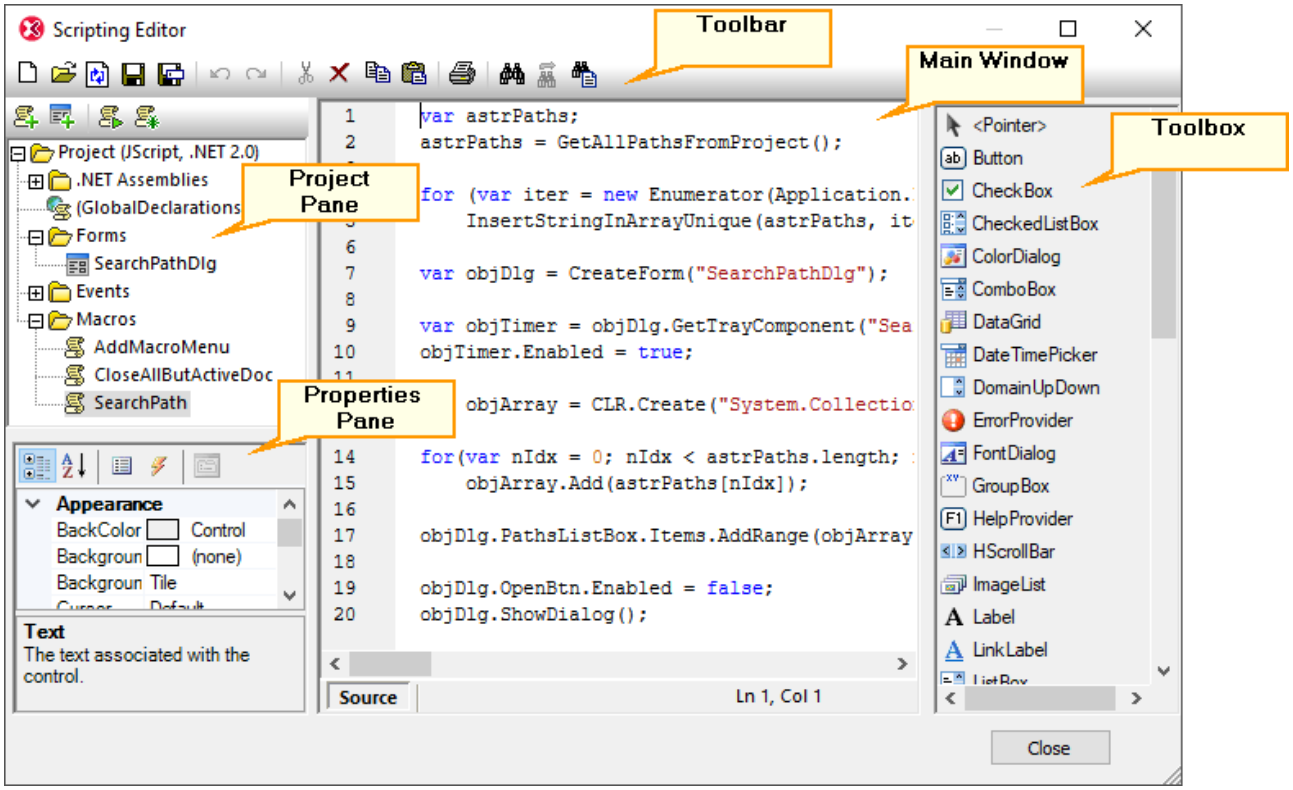
You can also find an example scripting project at the following path: **C:\Users\<user>\Documents\Altova\XMLSpy2021\Examples\SampleScripts.asprj**.

The next sections focus on the parts that your scripting project may need: global declarations, macros, forms, and events.

25.1.1.1 Overview of the Environment

The Scripting Editor consists of the following parts:

- Toolbar
- Project pane
- Properties pane
- Main window
- Toolbox



Toolbar

The toolbar includes standard Windows file management commands (**New**, **Open**, **Save**, **Save As**) commands and editor commands (**Copy**, **Cut**, **Delete**, **Paste**). When editing source code, the **Find** and **Replace** commands are additionally available, as well as the **Print** command.

Project pane

The project pane helps you view and manage the structure of the project. A scripting project consists of several components that can work together and may be created in any order:

- A *"Global Declarations"* script. As the name suggests, this script stores information available globally across the project. You can declare in this script any variables or functions that you need to be available in all forms, event handler scripts, and macros.
- *Forms.* Forms are typically necessary to collect user input, or provide some informative dialog boxes. A form is invoked by a call to it either within a function (in the Global Declarations script) or directly in a macro.
- *Events.* The "Events" folder displays XMLSpy application events provided by the COM API. To write a script that will be executed when an event occurs, double-click any event, and then type the handling code in the editor. The application events should not be confused with form events; the latter are handled at form level, as further detailed below.
- *Macros*. A macro is a script that can be invoked either on demand from a context menu or be executed automatically when XMLSpy starts. Macros do not have parameters or return values. A macro can access all variables and functions declared in the Global Declarations script and it can also display forms.

Right-click any of the components to see the available context menu commands and their shortcuts. Double-click any file (such as a form or a script) to open it in the main window.

The toolbar buttons provide the following quick commands:

	New macro	Adds a new macro to the project, in the **Macros** directory.
	New form	Adds a new form to the project, in the **Forms** directory.
	Run macro	Runs the selected macro.
	Debug macro	Runs the selected macro in debug mode.

Properties pane

The Properties pane is very similar to the one in Visual Studio. It displays the following:

- Form properties, when a form is selected
- Object properties, when an object in a form is selected
- Form events, when a form is selected
- Object events, when an object in a form is selected

To switch between the properties and events of the selected component, click the **Properties** or **Events** buttons, respectively.

The **Categorized** and **Alphabetical** icons display the properties or events either organized by category or organized in ascending alphabetical order.

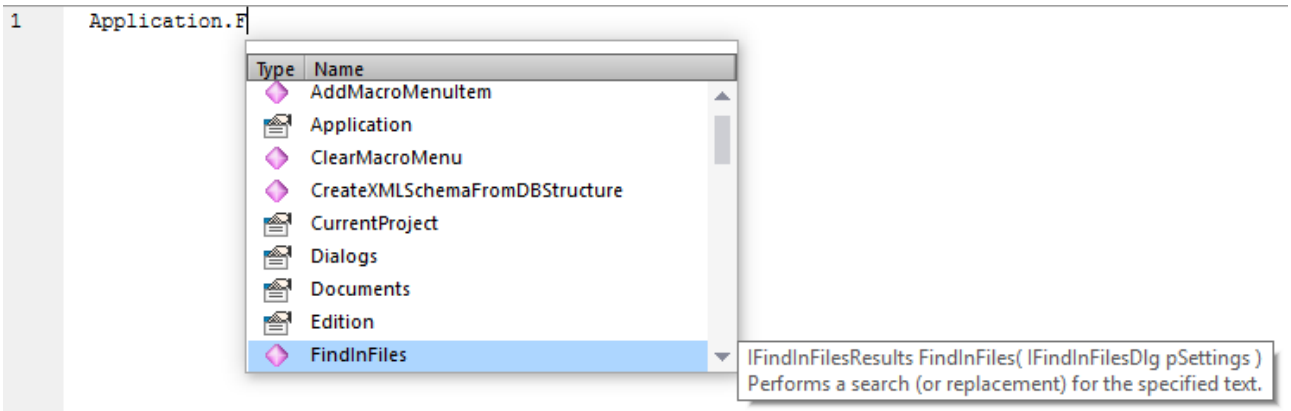
When a property or event is selected, a short description of it is displayed at the bottom of the Properties pane.

Main window

The main window is the working area where you can enter source code or modify the design of the form. When editing forms, you can work in two tabs: the **Design** tab and the **Source** tab. The **Design** tab shows the layout of the form, while the **Source** tab contains the source code such as handler methods for the form events.

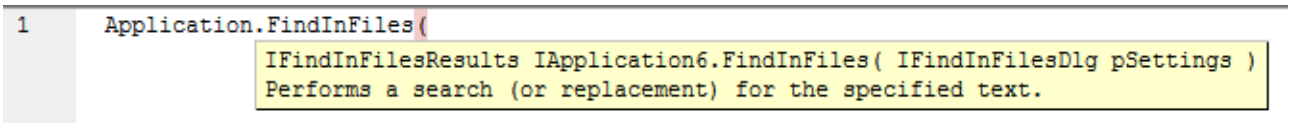
The source code editor provides code editing aids such as syntax coloring, source code folding, highlighting of starting and ending braces, zooming, autocompletion suggestions, bookmarks.

Autocompletion suggestions
JScript and VBScript are untyped languages, so autocompletion is limited to COM API names and XMLSpy built-in commands 1069. The full method or property signature is shown next to the autocompletion entry helper.



If names start with `objDocument`, `objProject`, `objXMLData`, or `objAuthenticRange`, members of the corresponding interface will be shown.

Placing the mouse over a known method or property displays its signature (and documentation if available), for example:



The auto-completion entry helper is normally shown automatically during editing, but it can also be obtained on demand by pressing **Ctrl+Space**.

Bookmarks

- To set or remove a bookmark, click inside a line, and then press **Ctrl+F2**
- To navigate to the next bookmark, press **F2**
- To navigate to the previous bookmark, press **Shift+F2**
- To delete all bookmarks, press **Ctrl+Shift+F2**

Zooming in/out

- To zoom in or out, hold the **Ctrl** key pressed and then press the "+" or "-" keys or rotate the mouse wheel.

Text view settings
To trigger text settings, right-click inside the editor, and select **Text View Settings** from the context menu.

Font settings
To change the font, right-click inside the editor, and select **Text View Font** from the context menu.

Toolbox

The Toolbox contains all the objects that are available for designing forms, such as buttons, text boxes, combo boxes, and so on.

To add a Toolbox item to a form:

1. Create or open a form and make sure that the **Design** tab is selected.
2. Click the Toolbox object (for example, **Button**), and then click at the location in the form where you wish to insert it. Alternatively, drag the object directly onto the form.

Some objects such as `Timer` are not added to the Form but are created in a tray at the bottom of the main window. You can select the object in the tray and set properties and event handlers for the object from the Properties pane. For an example of handling tray components from the code, see Handling form events 1063.

You can also add registered ActiveX controls to the form. To do this, right-click the Toolbox area and select **Add ActiveX Control** from the context menu.

25.1.1.2 Global Declarations

The "Global Declarations" script is present by default in any scripting project; you do not need to create it explicitly. Any variables or functions that you add to this script are considered global across the entire project. Consequently, you can refer to such variables and functions from any of the project's macros and events. The following is an example of a global declarations script that imports the `System.Windows.Forms` namespace into the project. To achieve that, the code below invokes the `CLR.Import` command built into Scripting Editor.

```
// import System.Windows.Forms namespace for all macros, forms and events:
CLR.Import( "System.Windows.Forms" );
```

Note: Every time a macro is executed or an event handler is called, the global declarations are re-initialized.

25.1.1.3 Macros

Macros are scripts that contain JScript (or VBScript, depending on your project's language) statements, such as variable declarations and functions.

If your projects should use macros, you can add them as follows: right-click inside the Project pane, select **Add Macro** from the context menu, and then enter the macro's code in the main form. The code of a macro could be as simple as an alert, for example:

```
alert("Hello, I'm a macro!");
```

More advanced macros can contain variables and local functions. Macros can also contain code that invokes forms from the project. The listing below illustrates an example of a macro that shows a form. It is assumed that this form has already been created in the "Forms" folder and has the name "SampleForm", see also Forms 1062.

```
// display a form
ShowForm( "SampleForm" );
```

In the code listing above, `ShowForm` is a command built into Scripting Editor. For reference to other similar commands that you can use to work with forms and .NET objects, the Built-in Commands 1069.

You can add multiple macros to the same project, and you can designate any macro as "auto-macro". When a macro is designated as "auto-macro", it runs automatically when XMLSpy starts. To designate a macro as auto-macro, right-click it, and select **Set as Auto-Macro** from the context menu.

Only one macro can be run at a time. After a macro (or event) is executed, the script is closed and global variables lose their values.

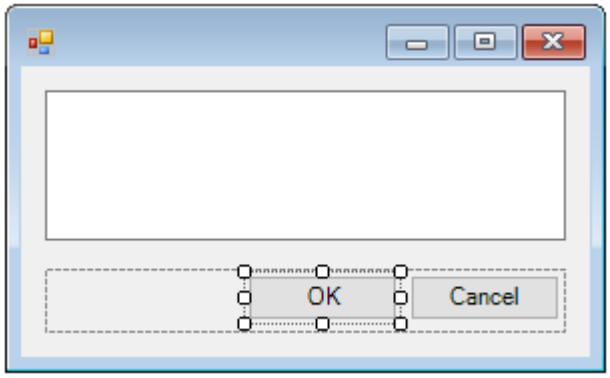
To run a macro directly in Script Editor, click **Run Macro** . To debug a macro using the Visual Studio debugger, click **Debug Macro** . For information about enabling and running macros in XMLSpy, see Enabling Scripts and Macros 1079.

25.1.1.4 Forms

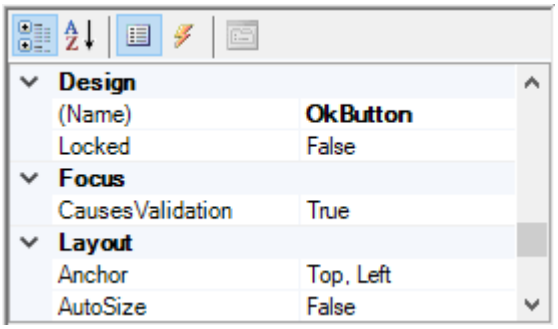
Forms are particularly useful if you need to collect input data from users or display data to users. A form can contain miscellaneous controls to facilitate this, such as buttons, check boxes, combo boxes, and so on.

To add a form, right-click inside the Project pane, and then select **Add Form** from the context menu. To add a control to a form, drag it from the Toolbox available to the right side of Scripting Editor and drop it onto the form.

You can change the position and size of the controls directly on the form, by using the handles that appear when you click any control, for example:



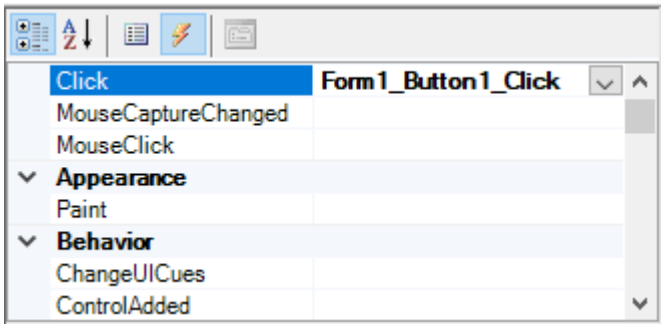
All form controls have properties that you can easily adjust in the Properties pane. To do this, first select the control on the form, and then edit the required properties in the Properties pane.



Handling form events

Each form control also exposes various events to which your scripting project can bind. For example, you might want to invoke some XMLSpy COM API method whenever a button is clicked. To create a function that binds to a form event, do the following:

1. In the Properties pane, click **Events** .
2. In the **Action** column, double-click the event where you need the method (for example, in the image below, the handled event is "Click").



You can also add handler methods by double-clicking a control on the form. For example, double-clicking a button in the form design generates a handler method for the "Click" event of that button.

Once the body of the handler method is generated, you can type code that handles this event, for example:

```
//Occurs when the component is clicked.
function MyForm_ButtonClick( objSender, e_EventArgs )
{
   alert("A button was clicked");
}
```

To display a work-in-progress form detached from the Scripting Editor, right-click the form, and select **Test Form** from the context menu. Note that the **Test Form** command just displays the form; the form's events (such as button clicks) are still disabled. To have the form react to events, call it from a macro, for example:

```
// Instantiate and display a form
ShowForm( "SampleForm" );
```

Accessing form controls

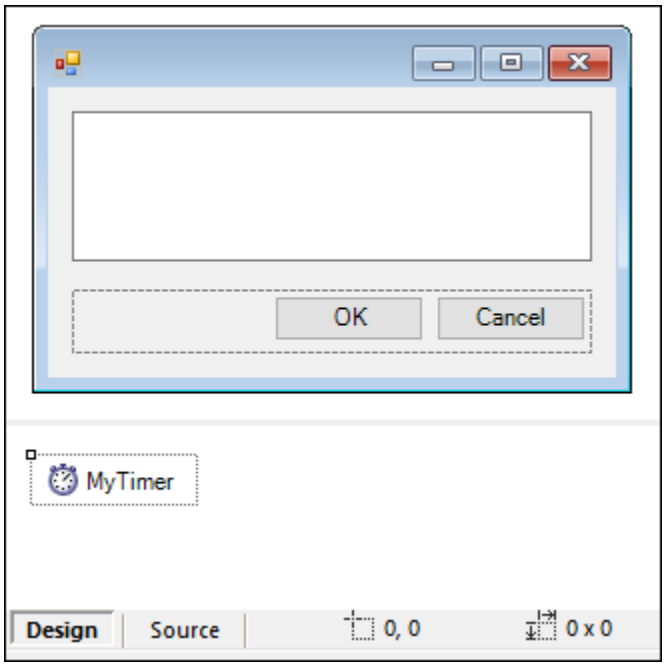
You can access any components on a form from your code by using field access syntax. For example, suppose there is a form designed as follows:

```
// MyForm
//   ButtonPanel
//      OkButton
//      CancelButton
//   TextEditor
//      AxMediaPlayer1
// TrayComponents
//   MyTimer
```

The code below shows how to instantiate the form, access some of its controls using field access syntax, and then display the form:

```
// Instantiate the form
var objForm = CreateForm("MyForm");
// Disable the OK button
objForm.ButtonPanel.OkButton.Enabled = false;
// Change the text of TextEditor
objForm.TextEditor.Text = "Hello";
// Show the form
objForm.ShowDialog();
```

When you add certain controls such as timers to the form, they are not displayed on the form; instead, they are shown as tray components at the base of the form design, for example:



To access controls from the tray, use the `GetTrayComponent` method on the form object, and supply the name of the control as argument. In this example, to get a reference to `MyTimer` and enable it, use the following code:

```
var objTimer = objForm.GetTrayComponent("MyTimer");
objTimer.Enabled = true;
```

For ActiveX Controls, you can access the underlying COM object via the OCX property:

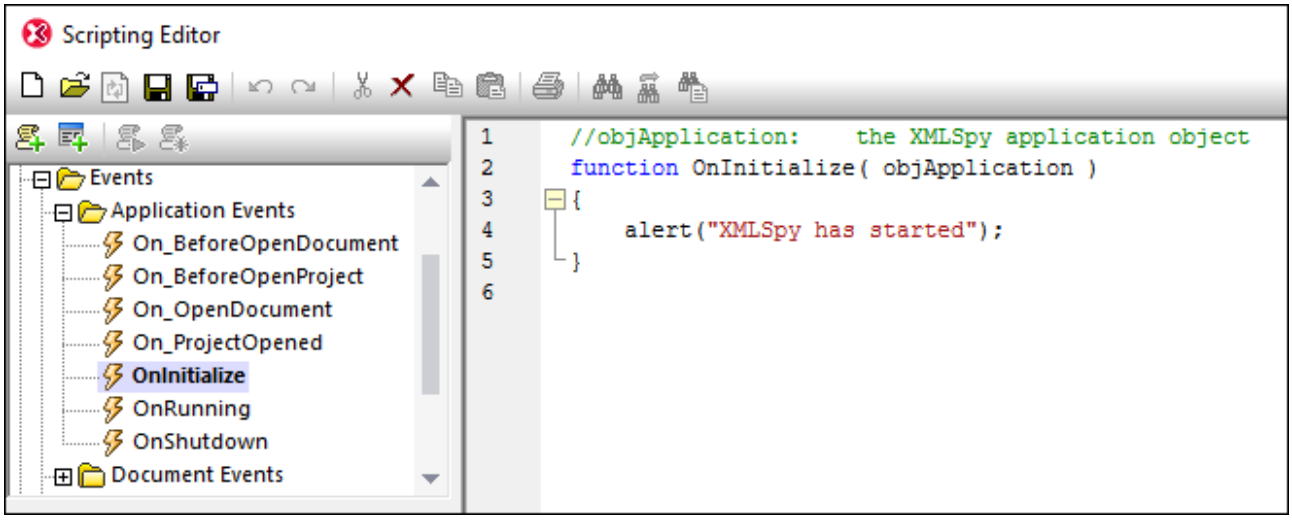
```
var ocx = lastform.AxMediaPlayer1.OCX; // get underlying COM object
ocx.enableContextMenu = true;
ocx.URL = "mms://apasf.apa.at/fm4_live_worldwide";
```

25.1.1.5 Events

Your scripting project may optionally include scripts that handle XMLSpy events such as opening, closing, or saving a document, starting or closing XMLSpy, adding an element to a diagram, and others. These events are provided by the XMLSpy COM API, and you can find them in the "Events" folder of your scripting project. Note that these events are XMLSpy-specific, as opposed to form events. Events are organized into folders as follows:

- Application Events
- Document Events
- AuthenticView Events
- GridView Events
- TextView Evets

To create an event handler script, right-click an event, and select **Open** from the context menu (or double-click the event). The event handler script is displayed in the main window, where you can start editing it. For example, the event handler illustrated below displays an alert each time XMLSpy starts:



Note the following:

- The `alert` command is applicable to JScript. The VBScript equivalent is `MsgBox`. See also alert[1070].
- The name of the event handler function must not be changed; otherwise, the event handler script will not be called.
- In order for events to be processed, the **Process Events** check box must be selected when you enable the scripting project in XMLSpy. For more information, see Enabling Scripts and Macros[1079].

You can optionally define local variables and helper functions within event handler scripts, for example:

```
var local;

function OnInitialize( objApplication )
{
   local = "OnInitialize";
   Helper();
}

function Helper()
{
   alert("I'm a helper function for " + local);
}
```

25.1.1.6 JScript Programming Tips

Below are a few JScript programming tips that you may find useful while developing a scripting project in XMLSpy Scripting Editor.

Out parameters

Out parameters from methods of the.NET Framework require special variables in JScript. For example:

```
var dictionary =
CLR.Create("System.Collections.Generic.Dictionary<System.String,System.String>");
dictionary.Add("1", "A");
dictionary.Add("2", "B");

// use JScript method to access out-parameters
var strOut = new Array(1);
if ( dictionary.TryGetValue("1", strOut) ) // TryGetValue will set the out parameter
  alert( strOut[0] ); // use out parameter
```

Integer arguments

.NET Methods that require integer arguments should not be called directly with JScript number objects which are floating point values. For example, instead of:

```
var objCustomColor = CLR.Static("System.Drawing.Color").FromArgb(128,128,128);
```

use:

```
var objCustomColor =
CLR.Static("System.Drawing.Color").FromArgb(Math.floor(128),Math.floor(128),Math.floor(
128));
```


(C) 2015-2021 Altova GmbH

Iterating .NET collections

To iterate .NET collections, the JScript Enumerator as well as the .NET iterator technologies can be used, for example:

```
// iterate using the JScript iterator
var itr = new Enumerator( coll );
for ( ; !itr.atEnd(); itr.moveNext() )
  alert( itr.item() );

// iterate using the .NET iterator
var itrNET = coll.GetEnumerator();
while( itrNET.MoveNext() )
  alert( itrNET.Current );
```

.NET templates

.NET templates can be instantiated as shown below:

```
var coll = CLR.Create( "System.Collections.Generic.List<System.String>" );
```

or

```
CLR.Import( "System" );
CLR.Import( "System.Collections.Generic" );
var dictionary = CLR.Create( "Dictionary<String,Dictionary<String,String>>" );
```

.NET enumeration values

.NET enumeration values are accessed as shown below:

```
var enumValStretch = CLR.Static( "System.Windows.Forms.ImageLayout" ).Stretch;
```

Enumeration literals

The enumeration literals from the XMLSpy API can be accessed as shown below (there is no need to know their numerical value).

```
objExportXMIFileDlg.XMIType = eXMI21ForUML23;
```

25.1.1.7 Example Scripting Project

A demo project that illustrates scripting with XMLSpy is available at the following path: **C:\Users\<user>\Documents\Altova\XMLSpy2021\Examples\SampleScripts.asprj**. This scripting project consists of a few macros and a Windows form.

To load the scripting project into Scripting Editor:

1. On the **Tools** menu, click **Scripting Editor**.
2. Click **Open** and browse for the **SampleScripts.asprj** file from the path above.

The project contains several macros in the "Macros" directory.

Macro	Description
AddMacroMenu	This macro adds a new menu item to XMLSpy, by invoking the `Application.AddMacroMenuItem` method of the COM API. The first argument of the `AddMacroMenuItem` method is the name of the macro to be added (in this example, "CloseAllButActiveDoc") and the second argument is the display text for the menu item. Whenever this macro is run, a new menu command called "CloseAllButActiveDoc") is added under the **Tools** menu. To clear macro menu items created previously, either restart XMLSpy or create a macro that calls the `Application.ClearMacroMenu` API method.
CloseAllButActiveDocument	When executed, the macro iterates though the currently open documents in XMLSpy and closes all of them, except for the active document.
SearchPath	This macro displays a form that lets users perform search for files within the current project. The form is available in the "Forms" directory, where you can view its design and the associated event handlers. The `GetAllPathsFromProject()` method returns all the file paths that belong to the currently opened project, as an array. The definition of this method is in the **GlobalDeclarations** script of the project. The `InsertStringInArrayUnique` method ensures that only unique paths are added to the array. Next, the form is initialized with CreateForm 1075. Finally, the array is converted to a .NET type with the help of the CLR.Create 1071 method and the form is populated with the resulting `ArrayList` collection. The **Open** button of the form has a handler that calls the `Application.Documents.OpenFile` API method to open the currently selected file.

To enable the scripting project as global XMLSpy scripting project:

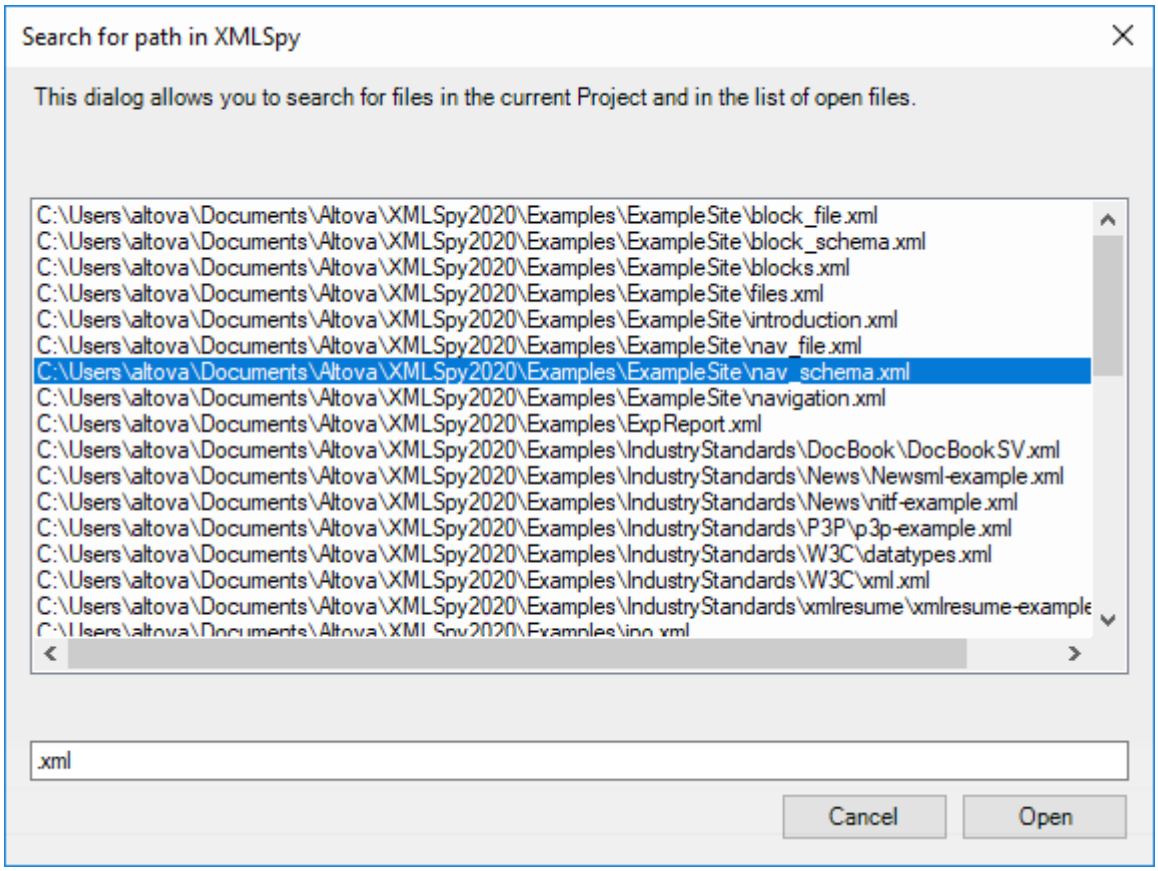
1. On the **Tools** menu, click **Options**.
2. Click the **Scripting** tab.
3. Under "Global scripting project file", click **Browse** and select the **SampleScripts.asprj** file from the path above.
4. This scripting project does not have auto-macros and application event handlers; therefore, you don't need to select either the **Run auto-macros...** or **Process events** check boxes.
5. Click **Apply**.

(C) 2015-2021 Altova GmbH

At this stage, several new menu items (one for each macro) become available under the **Tools | Macros** menu.

To run the "SearchPath" macro:

1. Open an XMLSpy project that contains several files (in this example, **C:\Users\<user>\Documents\Altova\XMLSpy2021\Examples\Examples.spp**).
2. On the **Tools** menu, click **Macros**, and then click **Search Path**.
3. Type the search term (in this example, ".xml").



As shown above, all file names that contain the search term are now listed. You can click any element in the list, and then click **Open** to display it in the main editor.

25.1.2 Built-in Commands

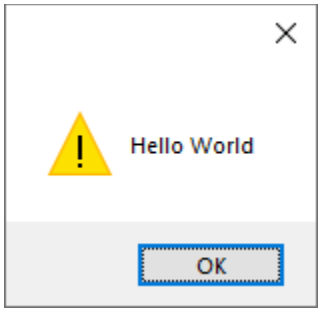
This section provides reference to all the commands you can use in the XMLSpy Scripting Editor.

- alert 1070
- confirm 1070
- CLR.Create 1071
- CLR.Import 1072
- CLR.LoadAssembly 1072
- CLR.ShowImports 1073

- CLR.ShowLoadedAssemblies 1074
- CLR.Static 1074
- CreateForm 1075
- doevents 1076
- lastform 1076
- prompt 1077
- ShowForm 1077
- watchdog 1078

25.1.2.1 alert

Displays a message box that shows a given message and the "OK" button. To proceed, the user will have to click "OK".



Signature

For JScript, the signature is:

```
alert(strMessage : String) -> void
```

For VBScript, the signature is:

```
MsgBox(strMessage : String) -> void
```

Example

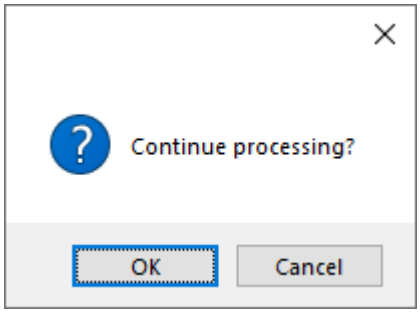
The following JScript code displays a message box with the text "Hello World".

```
alert("Hello World");
```

25.1.2.2 confirm

Opens a dialog box that shows a given message, a confirmation button, and a cancel button. The user will have to click either "OK" or "Cancel" to proceed. Returns a Boolean that represents the user's answer. If the user clicked "OK", the function returns **true**; if the user clicked "Cancel", the function returns **false**.

(C) 2015-2021 Altova GmbH



Signature

```
confirm(strMessage : String) -> result : Boolean
```

Example (JScript)

```
if ( confirm( "Continue processing?" ) == false )
  alert("You have cancelled this action");
```

Example (VBScript)

```
If ( confirm( "Continue processing?" ) = false ) Then
  MsgBox("You have cancelled this action")
End If
```

25.1.2.3 CLR.Create

Creates a new .NET object instance of the type name supplied as argument. If more than one argument is passed, the successive arguments are interpreted as the arguments for the constructor of the .NET object. The return value is a reference to the created .NET object

Signature

```
CLR.Create(strTypeNameCLR : String, constructor arguments ... ) -> object
```

Example

The following JScript code illustrates how to create instances of various .NET classes.

```
// Create an ArrayList
var objArray = CLR.Create("System.Collections.ArrayList");
// Create a ListViewItem
var newItem = CLR.Create( "System.Windows.Forms.ListViewItem", "NewItemText" );
// Create a List<string>
var coll = CLR.Create( "System.Collections.Generic.List<System.String>" );
// Import required namespaces and create a Dictionary object
```

```
CLR.Import( "System" );
CLR.Import( "System.Collections.Generic" );
var dictionary = CLR.Create( "Dictionary< String, Dictionary< String, String > >" );
```

25.1.2.4 CLR.Import

Imports a namespace. This is the scripting equivalent of C# `using` and VB.Net `imports` keyword. Calling `CLR.Import` makes it possible to leave out the namespace part in subsequent calls like `CLR.Create()` and `CLR.Static()`.

Note: Importing a namespace does not add or load the corresponding assembly to the scripting project. You can add assemblies to the scripting project dynamically (at runtime) in the source code by calling CLR.LoadAssembly[1072].

Signature

```
CLR.Import(strNamespaceCLR : String) -> void
```

Example

Instead of having to use fully qualified namespaces like:

```
if ( ShowForm( "FormName" ) == CLR.Static( "System.Windows.Forms.DialogResult" ).OK )
{
    var sName = lastform.textboxFirstName.Text + " " + lastform.textboxLastName.Text;
    CLR.Static( "System.Windows.Forms.MessageBox" ).Show( "Hello " + sName );
}
```

One can import namespaces first and subsequently use the short form:

```
CLR.Import( "System.Windows.Forms" );

if ( ShowForm( "FormName" ) == CLR.Static( "DialogResult" ).OK )
{
    var sName = lastform.textboxFirstName.Text + " " + lastform.textboxLastName.Text;
    CLR.Static( "MessageBox" ).Show( "Hello " + sName );
}
```

25.1.2.5 CLR.LoadAssembly

Loads the .NET assembly with the given long assembly name or file path. Returns Boolean **true** if the assembly could be loaded; **false** otherwise.

Signature

```
CLR.LoadAssembly(strAssemblyNameCLR : String, showLoadErrors : Boolean) -> result :
Boolean
```


(C) 2015-2021 Altova GmbH

Example

The following JScript code attempts to set the clipboard text by loading the required assembly dynamically.

```
// set clipboard text (if possible)
// System.Windows.Clipboard is part of the PresentationCore assembly, so load this
assembly first:
if ( CLR.LoadAssembly( "PresentationCore, Version=3.0.0.0, Culture=neutral,
PublicKeyToken=31bf3856ad364e35", true ) )
{
    var clipboard = CLR.Static( "System.Windows.Clipboard" );
    if ( clipboard != null )
        clipboard.SetText( "HelloClipboard" );
}
```

25.1.2.6 CLR.ShowImports

Opens a message box that shows the currently imported namespaces. The user will have to click "OK" to proceed.

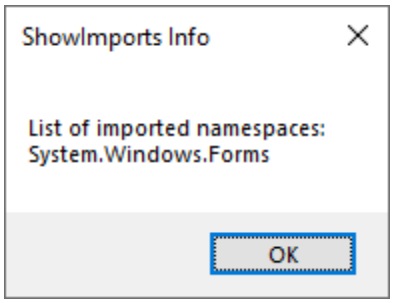
Signature

```
CLR.ShowImports() -> void
```

Example

The following JScript code first imports a namespace, and then displays the list of imported namespaces:

```
CLR.Import( "System.Windows.Forms");
CLR.ShowImports();
```



25.1.2.7 CLR.ShowLoadedAssemblies

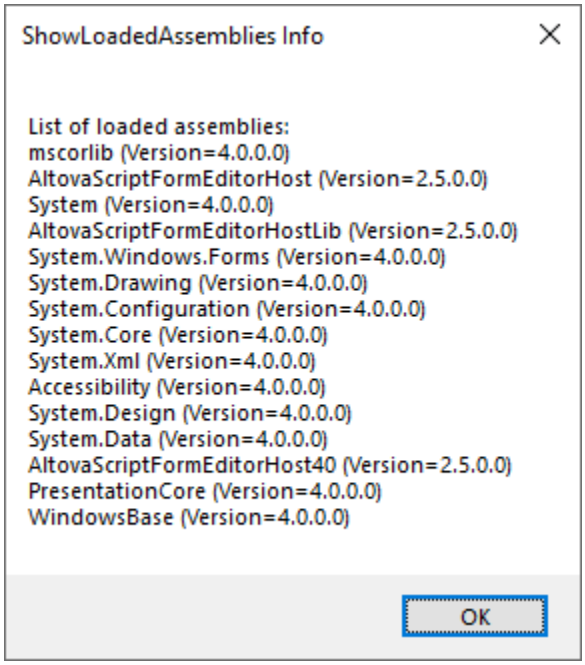
Opens a message box that shows the currently loaded assemblies. The user will have to click "OK" to proceed.

Signature

```
CLR.ShowLoadedAssemblies() -> void
```

Example

```
CLR.ShowLoadedAssemblies();
```



25.1.2.8 CLR.Static

Returns a reference to a static .NET object. You can use this function to get access to .NET types that have no instances and contain only static members.

Signature

```
CLR.Static(strTypeNameCLR : String) -> object
```

Example (JScript)

```
// Get the value of a .NET Enum into a variable
```

```
var enumValStretch = CLR.Static( "System.Windows.Forms.ImageLayout" ).Stretch

// Set the value of the Windows clipboard
var clipboard = CLR.Static( "System.Windows.Clipboard" );
clipboard.SetText( "HelloClipboard" );

// Check the buttons pressed by the user on a dialog box
if ( ShowForm( "FormName" ) == CLR.Static( "System.Windows.Forms.DialogResult" ).OK )
    alert( "ok" );
else
    alert( "cancel" );
```

25.1.2.9 CreateForm

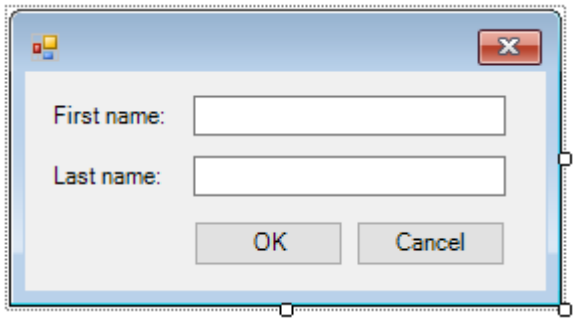
Instantiates the `Form` object identified by the name supplied as argument. The form must exist in the "Forms" folder of the scripting project. Returns the form object (`System.Windows.Forms.Form`) corresponding to the given name, or `null` if no form with such name exists.

Signature

```
CreateForm (strFormName : String) -> System.Windows.Forms.Form | null
```

Example

Let's assume that a form called "FormName" exists in the scripting project.



The following JScript code instantiates the form with some default values and displays it to the user.

```
var myForm = CreateForm( "FormName" );
if ( myForm != null )
{
   myForm.textboxFirstName.Text = "Daniela";
   myForm.textboxLastName.Text = "Heidegger";
   var dialogResult = myForm.ShowDialog();
}
```

The `dialogResult` can subsequently be evaluated as follows:

```
if ( dialogResult == CLR.Static( "System.Windows.Forms.DialogResult" ).OK )
```

```
    alert( "ok" );
else
    alert( "cancel" );
```

Note: The code above will work only if the **DialogResult** property of the "OK" and "Cancel" buttons is set correctly from the Properties pane (for example, it must be **OK** for the "OK" button).

25.1.2.10 doevents

Processes all Windows messages currently in the message queue.

Signature

```
doevents() -> void
```

Example (JScript)

```
for ( i=0; i < nLongLastingProcess; ++i )
{
    // do long lasting process

    doevents(); // process Windows messages; give UI a chance to update
}
```

25.1.2.11 lastform

This is a global field that returns a reference to the last form object that was created via `CreateForm()` or `ShowForm()`.

Signature

```
lastform -> formObj : System.Windows.Forms.Form
```

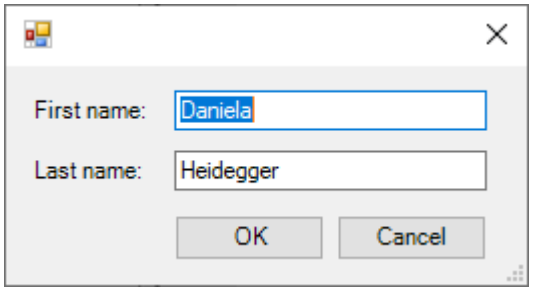
Example

The following JScript code shows the form "FormName" as a dialog box.

```
CreateForm( "FormName" );
if ( lastform != null )
{
    lastform.textboxFirstName.Text = "Daniela";
    lastform.textboxLastName.Text = "Heidegger";
    var dialogResult = lastform.ShowDialog();
}
```

The values of both textbox controls are initialized with the help of `lastform`.

(C) 2015-2021 Altova GmbH



25.1.2.12 prompt

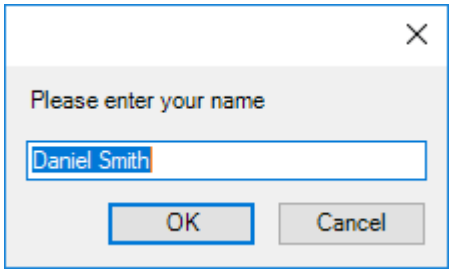
Opens a dialog box that shows a message and a textbox control with a default answer. This can be used to let the user input a simple string value. The return value is a string that contains the textbox value or null if the user selected "Cancel".

Signature

```
prompt(strMessage : String, strDefault : String) -> val : String
```

Example

```
var name = prompt( "Please enter your name", "Daniel Smith" );
if ( name != null )
    alert( "Hello " + name + "!" );
```



25.1.2.13 ShowForm

Instantiates a new form object from the given form name and immediately shows it as dialog box. The return value is an integer that represents the generated `DialogResult` (`System.Windows.Forms.DialogResult`). For the list of possible values, refer to the documentation of the `DialogResult` Enum (https://docs.microsoft.com/en-us/dotnet/api/system.windows.forms.dialogresult?view=netframework-4.8).

Signature

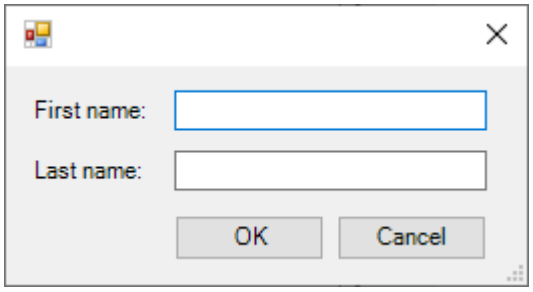
```
ShowForm(strFormName : String) -> result : Integer
```

Example

The following JScript code

```
var dialogResult = ShowForm( "FormName" );
```

Shows the form "FormName" as a dialog box:



The `DialogResult` can subsequently be evaluated, for example:

```
if ( dialogResult == CLR.Static( "System.Windows.Forms.DialogResult" ).OK )
    alert( "ok" );
else
    alert( "cancel" );
```

Note: The code above will work only if the **DialogResult** property of the "OK" and "Cancel" buttons is set correctly from the Properties pane (for example, it must be **OK** for the "OK" button).

25.1.2.14 watchdog

Long running CPU-intensive scripts may ask the user if the script should be terminated. The `watchdog()` method is used to disable or enable this behavior. By default, the watchdog is enabled.

Calling `watchdog(true)` can also be used to reset the watchdog. This can be useful before executing long running CPU-intensive tasks to ensure they have the maximum allowed script processing quota.

Signature

```
watchdog(bEnable : boolean) -> void
```

Example

```
watchdog( false );  // disable watchdog - we know the next statement is CPU intensive
but it will terminate for sure
doCPUIntensiveScript();
watchdog( true );   // re-enable watchdog
```


(C) 2015-2021 Altova GmbH

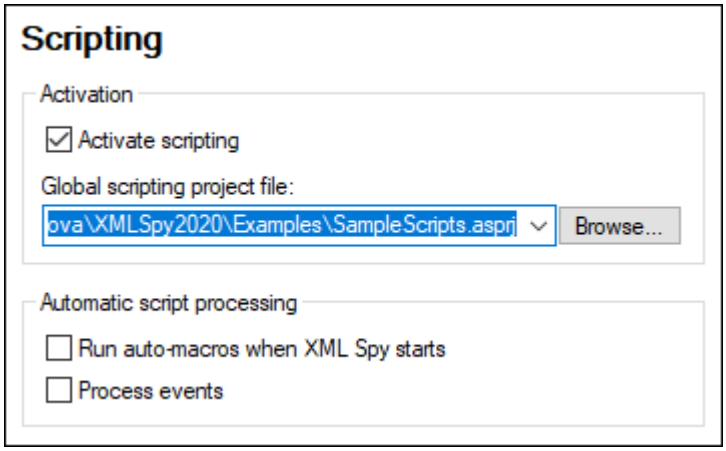
25.1.3 Enabling Scripts and Macros

Once a scripting project is complete and tested, you can use it in the following ways:

1. As the global scripting project for XMLSpy. This means that all the scripts and macros from the scripting project are available to XMLSpy.
2. At project level. This means that a reference to the .asprj file is saved together with the XMLSpy project. When the XMLSpy project is opened, its associated scripts and macros can be called.

To set a scripting project as global:

1. On the **Tools** menu, click **Options**.
2. Click the **Scripting** tab.
3. Select the **Activate scripting** check box and browse for the .asprj file to be used as global scripting project.



You can optionally enable the following additional script processing options:

Run auto-macros when XMLSpy starts	If you select this check box, any macros that were set as "Auto-macro" in the project will be triggerred automatically when XMLSpy starts.
Process events	Select this check box if your scripts bind to any application events. Clear the check box to prevent the scripts from reacting to events.

To enable a scripting project at project level:

1. Open the project.
2. On the **Project** menu, click **Script Settings**.
3. Select the **Activate project scripts** check box and browse for the .asprj file.

The **Run-auto macros...** check box has the same meaning as already described above.

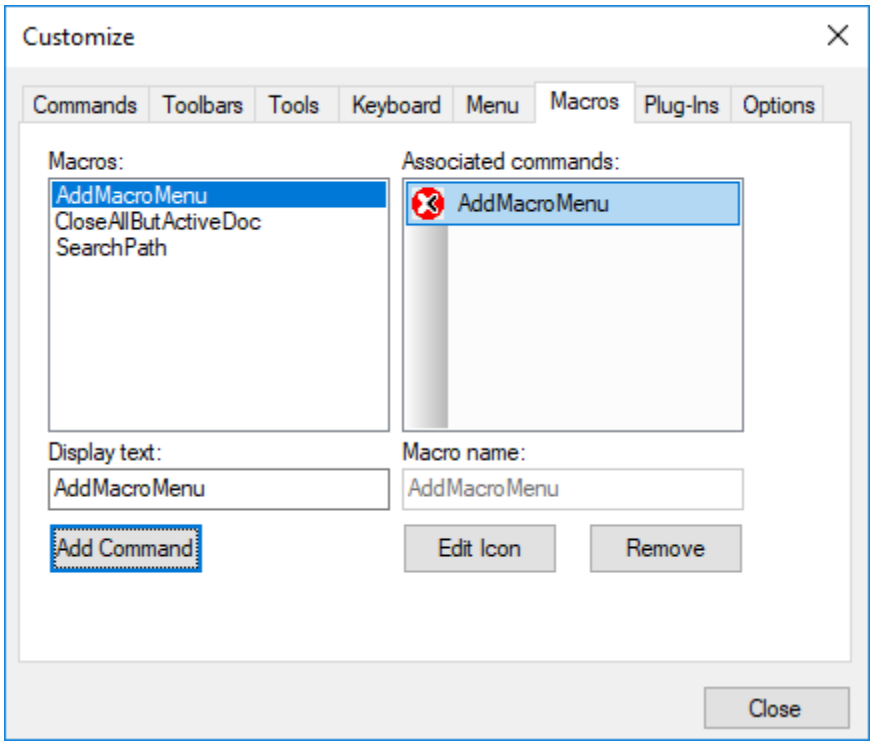
25.1.3.1 Running Macros

When a scripting project is active in XMLSpy, any macros available in that project are displayed in the **Tools | Macros** menu. Therefore, you can run a macro at any time, by triggering the respective menu command, for example **Tools | Macros | <SomeMacro>**.

Macros that were configured as auto-macros will run automatically whenever XMLSpy starts, provided that this behavior is enabled from options, as described in Enabling Scripts and Macros 1079.

For convenience, you can create toolbar buttons for macros, as follows:

1. On the **Tools** menu, click **Customize**.
2. Click the **Macros** tab. Any macros that are available at application level (in the *global* scripting project) are listed.
3. Click **Add Command**.



4. Optionally, click **Edit icon** and draw a new icon for the new macro. You can also assign a shortcut to the macro, from the **Keyboard** tab.
5. Drag the macro from the **Associated commands** pane onto the toolbar where you would like it to appear.

To remove a macro from a toolbar:

1. On the **Tools** menu, click **Customize**.
2. Click the **Macros** tab.
3. Drag the macro from the toolbar where it appears back into the **Associated commands** pane.

(C) 2015-2021 Altova GmbH

25.2 IDE Plugins

XMLSpy allows you to create your own IDE plug-ins and integrate them into XMLSpy.

Use plug-ins to:

- Configure your version of XMLSpy, add commands through menus, icons, buttons etc.
- React to events from XMLSpy.
- Run your specific code within XMLSpy with access to the complete XMLSpy API

XMLSpy expects your plug-in to implement the IXMLSpyPlugIn 1090 interface. C# and C++ are the currently supported languages, and examples using these languages are included with your installation package and are located in the `XMLSpy2021\Examples\IDEPlugin` folder of your XMLSpy installation.

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

See ATL sample files 1085 for an example using C++.

25.2.1 Registration of IDE PlugIns

XMLSpy maintains a specific key in the Registry where it stores all registered IDE plug-ins:

```
HKEY_CURRENT_USER\Software\Altova\XML Spy\PlugIns
```

All values of this key are treated as references to registered plug-ins and must conform to the following format:

Value name:	ProgID of the plug-in
Value type:	must be REG_SZ
Value data:	CLSID of the component

Each time the application starts the values of the **`PlugIns`** key is scanned, and the registered plug-ins are loaded.

Register plug-in manually

To register a plug-in manually, use the Customize dialog box of XMLSpy's **Tools** menu. Use the **Add Plug-In** button to specify the DLL that implements your plug-in. XMLSpy registers the DLL as a COM server and adds the corresponding entry in its **`PlugIns`** key.

If you experience problems with manual registration, check whether the CLSID of your plug-in is correctly registered in the **`PlugIns`** key. If the registration is incorrect, then the name of your plug-in DLL was probably not sufficiently unique. Use a different name or perform direct registration.

Register plug-in directly

A plug-in can be directly registered as an IDE plug-in by first registering the DLL and then adding the appropriate value to the **`PlugIns`** key of XMLSpy. (This can be done, for eaxmple, during plug-in setup.) The new plug-in will be activated the next time XMLSpy is launched.

Creating plug-ins

Source code for sample plug-ins has been provided in the application's (My) Documents folder [27]: `Examples\IDEPlugin` folder. To build a plug-in from such source code, do the following:

1. Open the solution you want to build as a plug-in in Visual Studio.
2. Build the plug-in with the command in the Build menu.
3. The plug-in's DLL file will be created in the `Bin` or `Debug` folder. This DLL file is the file that must be added as a plug-in (*see above*).

Note: C# and C++ are the currently supported languages.

25.2.2 ActiveX Controls

ActiveX controls are supported. Any IDE PlugIn which is also an ActiveX control will be displayed in a Dialog Control Bar. A sample PlugIn that is also an ActiveX control is included in the `IDEPlugin` folder in the `Examples` folder of your application folder.

25.2.3 Configuration XML

The IDE plug-in allows you to change the user interface (UI) of XMLSpy. This is done by describing each separate modification using an XML data stream. The XML configuration is passed to XMLSpy using the GetUIModifications [1093] method of the IXMLSpyPlugIn interface.

The XML file containing the UI modifications for the IDE PlugIn, must have the following structure:

```
<ConfigurationData>
   <ImageFile>Path to image file</ImageFile>
   <Modifications>
      <Modification>
         ...
      </Modification>
         ...
    </Modifications>
</ConfigurationData>
```

You can define icons or toolbar buttons for new menu items that are added to the UI of XMLSpy by the plug-in. The path to the file containing the images is set using the `ImageFile` element. Each image must be 16x16 pixels using maximum 256 colors. The image references must be arranged from left to right in a single **`<ImageFile>`** element. The rightmost image index value is zero.

The **`Modifications`** element can have any number of **`Modification`** child elements. Each **`Modification`** element defines a specific change to the standard UI of XMLSpy. The modifications you can carry out are described in the next section below.

Structure of Modification elements

A `Modification` element has two child elements:

```
<Modification>
```

 (C) 2015-2021 Altova GmbH

```
   <Action>Type of action</Action>
   <UIElement Type="Type of UI element" />
</Modification>
```

Valid values for the **Action** element are:

Add: to add the following UI element to XMLSpy
Hide: to hide the following UI element in XMLSpy
Remove: to remove the UI element from the "Commands" list box, in the customize dialog

Multiple modifications can be combined in an **Action** element, like this: **"Add Hide"**

The **UIElement** element defines any new or existing UI element and may be one of the the following types: toolbars, buttons, menus, or menu items. The **type** attribute specifies which of these types the UI element belongs to. The structure of **UIElement** is described in the sections below.

Common UIElement children

The `ID` and `Name` elements are defined for all types of UI element. In the case of some types, however, one of these elements is ignored. For example, **Name** is ignored for a separator.

```
<ID></ID>
<Name></Name>
```

If **UIElement** describes an existing element of the UI, the value of the **ID** element is predefined by XMLSpy. Normally these ID values are not known to the public. If the XML fragment describes a new part of the UI, then the ID is arbitrary and the value should be less than **1000**. The **Name** element sets the textual value. Existing UI elements can be identified just by name; for example, menus and menu items that have sub menus. For new UI elements, the **Name** element sets the caption (for example, the title of a toolbar) or the text of a menu item.

Toolbars and Menus

To define a toolbar it is necessary to specify the ID and/or the name of the toolbar. An existing toolbar can be specified using only the name or ID (if the latter is known). To create a **new** toolbar, both values must be set. The **type** attribute must have a value of **ToolBar**.

```
<UIElement Type="ToolBar">
   <ID>1</ID>
   <Name>TestPlugIn</Name>
</UIElement>
```

To specify an XMLSpy menu you need two parameters:

- The ID of the menu bar which contains the menu. If no XML documents are open in the main window, the menu bar ID is `128`. If one or more XML documents are open, the menu bar ID is `129`.
- The menu name. Menus do not have an associated ID value. The following example defines the "Edit" menu of the menu bar which is active, when at least one XML document is open:

  ```
  <UIElement Type="Menu">
     <ID>129</ID>
     <Name>Edit</Name>
  </UIElement>
  ```

An additional element is used if you want to create a new menu. The **Place** element defines the position of the new menu in the menu bar:

```
<UIElement Type="Menu">
   <ID>129</ID>
   <Name>PlugIn Menu</Name>
   <Place>12</Place>
</UIElement>
```

A value of `-1` for the `Place` element sets the new button or menu item at the end of the menu or toolbar.

Commands

If you add a new command (through a toolbar button or a menu item), the **`UIElement`** fragment can contain any of these sub elements:

```
   <MacroName></MacroName>
   <Info></Info>
   <ImageID></ImageID>
```

If **`MacroName`** is specified, XMLSpy searches for a macro with the same name in the scripting environment and executes it each time this command is processed. The **`Info`** element contains a description string that is displayed in the status bar when the mouse pointer is over the associated command (button or menu item). **`ImageID`** defines the index of the icon in the image file. Note that all icons are stored in one image file.

To define a toolbar button, create an `UIElement` with this structure:

```
<UIElement Type="ToolBarItem">
   <!--don't reuse local IDs even the commands do the same-->
   <ID>5</ID>
   <Name>Open file from repository...</Name>
   <!--Set Place To -1 If this is the first button To be inserted-->
   <Place>-1</Place>
   <ImageID>0</ImageID>
   <ToolBarID>1</ToolBarID>
   <!--instead of the toolbar ID the toolbar name could be used-->
   <ToolBarName>TestPlugIn</ToolBarName>
</UIElement>
```

Additional elements to declare a toolbar button are **`Place`**, **`ToolBarID`** and **`ToolBarName`**. The `ToolBarID` and `ToolBarName` elements are used to identify the toolbar which contains the new or existing button. The textual value of **`ToolBarName`** is case-sensitive. The (UIElement) **`type`** attribute must be **`ToolBarItem`**.

To define a menu item, the elements **`MenuID`**, **`Place`** and **`Parent`** are available in addition to the standard elements used to declare a command. The content of the **`MenuID`** element can be either **`128`** or **`129`**. *See the section "Toolbars and Menus" above for more information.*

The **`Parent`** element is used to identify the menu where the new menu entry should be inserted. As sub menu items have no unique Windows ID, we need some other way to identify the parent of the menu item. We do this by setting the content of the **`Parent`** element to be the path to the menu item. The steps in the path are indicated by a colon. The pattern would be **`ParentMenu:SubMenu`**. If the menu has no parent (because it is not a submenu), add a colon to the beginning of the name (see example below). The **`type`** attribute must be set to **`MenuItem`**.

The example below defines a menu item, where the containing menu is not a sub menu:

```
<UIElement Type="MenuItem">
   <!--the following element is a Local command ID-->
   <ID>3</ID>
   <Name>Open file from repository...</Name>
```

```
   <Place>-1</Place>
   <MenuID>129</MenuID>
   <Parent>:PlugIn Menu</Parent>
   <ImageID>0</ImageID>
</UIElement>
```

You can add toolbar separators and menus if the value of the `ID` element is set to `0`.

25.2.4 ATL Sample Files

This section shows how to create a simple XMLSpy IDE plug-in DLL using ATL. You must know how to work with MS VisualStudio, ATL, and the wizards that generate new ATL objects. To access the API, the implementation imports the Type Library of XMLSpy. The code reads various properties and calls methods using the smart pointers provided by the **`#import`** statement of the code. In addition, the sample code uses the MFC class **`CString`** and ATL conversion macros such as W2T.

The broad steps to create an ATL DLL are as follows:

1. Open VisualStudio and select **File | New**.
2. Select the *Projects* tab.
3. Select *ATL COM AppWizard*, and type in a project name.
4. Select *Support for MFC* if you want to use MFC classes or if you want to create a project for the sample code.

Having created the project files you can add an ATL object to implement the **`IXMLSpyPlugIn`** interface:

1. Select **Insert | New ATL Object**.
2. Select *Simple Object* from the wizard. and click **Next**.
3. Type in a name for the object.
4. On the *Attributes* tab, select *Custom* for the type of interface and disable *Aggregation*.

These steps produce the skeleton code for the implementation of the IDE plug-in interface. See the following pages for information about how to modify the code and specify some basic functionality.

25.2.4.1 Interface description (IDL)

The IDL of the newly created ATL object contains a declaration for one COM interface.

- This interface declaration must be replaced by the declaration of `IXMLSpyPlugIn` as shown below.
- The IDL must also contain the definition of the `SPYUpdateAction` enumeration.
- Replace the generated default interface name (created by the wizard) with `IXMLSpyPlugIn` in the `coclass` declaration.

The IDL should then look something like the example code below. After creating the ATL object, you need to implement the IDE plug-in interface of XMLSpy.

```
import "oaidl.idl";
import "ocidl.idl";
```

```
// ----- please insert the following block into your IDL file -----
   typedef enum {
      spyEnable = 1,
      spyDisable = 2,
      spyCheck = 4,
      spyUncheck = 8
   } SPYUpdateAction;

// ----- end insert block ----


// ----- E.g. Interface entry automatically generated by the ATL wizard -----
//     [
//           object,
//           uuid(AB7CD86A-8145-429A-A1F3-270692EO8AFC),

//           helpstring("IXMLSpyPlugIn Interface")
//           pointer_default(unique)
//     ]
//     interface IXMLSpyPlugIn : IUnknown
//     {
//     };
// ----- end automatically generated Interface Entry


// ----- replace the Interface Entry (shown above) generated for you by the ATL wizard,
with the following block -----

       [
          odl,
          uuid(88F2A622-4B7E-42CD-8D04-3C0E5389DD85),
          helpstring("IXMLSpyPlugIn Interface")
       ]
       interface IXMLSpyPlugIn : IUnknown
          {
             HRESULT _stdcall OnCommand([in] long nID, [in] IDispatch* pXMLSpy);
             HRESULT _stdcall OnUpdateCommand([in] long nID, [in] IDispatch* pXMLSpy,
[out, retval] SPYUpdateAction* pAction);
             HRESULT _stdcall OnEvent([in] long nEventID, [in] SAFEARRAY(VARIANT)*
arrayParameters, [in] IDispatch* pXMLSpy, [out, retval] VARIANT* pReturnValue);
             HRESULT _stdcall GetUIModifications([out, retval] BSTR* pModificationsXML);
             HRESULT _stdcall GetDescription([out, retval] BSTR* pDescription);
          };

// ----- end replace block -----


// ----- The code below is automatically generated by the ATL wizard and will look
slightly different in your case -----

       [
          uuid(24FE0D1B-3FC0-494E-B36E-1D4CE412B014),
          version(1.0),
          helpstring("XMLSpyIDEPlugInDLL 1.0 Type Library")
       ]
       library XMLSPYIDEPLUGINDLLLib
          {
       importlib("stdole32.tlb");
       importlib("stdole2.tlb");

       [
          uuid(3800E791-7F6B-4ACD-9E32-2AC184444501),
```

```
         helpstring("XMLSpyIDEPlugIn Class")
      ]
      coclass XMLSpyIDEPlugIn
      {
         [default] interface IXMLSpyPlugIn;   // ----- define IXMLSpyPlugIn as the
default interface -----
      };
};
```

25.2.4.2 Class definition

In the class definition of the ATL object, the following changes must be made:

- The class has to derive from `IXMLSpyPlugIn`
- The "Interface Map" needs an entry for `IXMLSpyPlugIn`
- The methods of the IDE plug-in interface must be declared

These changes can be made as shown below:

```
#ifndef __XMLSPYIDEPLUGIN_H_
#define __XMLSPYIDEPLUGIN_H_

#include "resource.h"       // main symbols

/////////////////////////////////////////////////////////////////////////////
// CXMLSpyIDEPlugIn
class ATL_NO_VTABLE CXMLSpyIDEPlugIn :
   public CComObjectRootEx<CComSingleThreadModel>,
   public CComCoClass<CXMLSpyIDEPlugIn, &CLSID_XMLSpyIDEPlugIn>,
   public IXMLSpyPlugIn
{
public:
   CXMLSpyIDEPlugIn()
   {
   }

DECLARE_REGISTRY_RESOURCEID(IDR_XMLSPYIDEPLUGIN)
DECLARE_NOT_AGGREGATABLE(CXMLSpyIDEPlugIn)

DECLARE_PROTECT_FINAL_CONSTRUCT()

BEGIN_COM_MAP(CXMLSpyIDEPlugIn)
   COM_INTERFACE_ENTRY(IXMLSpyPlugIn)
END_COM_MAP()

// IXMLSpyIDEPlugIn
public:
   virtual HRESULT _stdcall OnCommand(long nID, IDispatch* pXMLSpy);
   virtual HRESULT _stdcall OnUpdateCommand(long nID, IDispatch* pXMLSpy,
SPYUpdateAction* pAction);
   virtual HRESULT _stdcall OnEvent(long nEventID, SAFEARRAY **arrayParameters,
IDispatch* pXMLSpy, VARIANT* pReturnValue);
   virtual HRESULT _stdcall GetUIModifications(BSTR* pModificationsXML);
   virtual HRESULT _stdcall GetDescription(BSTR* pDescription);
};
```

```
#endif //__XMLSPYIDEPLUGIN_H_
```

25.2.4.3 Implementation

The code below shows a simple implementation of an XMLSpy IDE plug-in. It adds a menu item and a separator (available with XMLSpy) to the Tools menu. Inside the OnUpdateCommand() method, the new command is only enabled when the active document is displayed using the Grid View. The command searches for the XML element which has the current focus, and opens any URL starting with "http://", from the textual value of the element.

```
/////////////////////////////////////////////////////////////////////////////
// CXMLSpyIDEPlugIn

#import "XMLSpy.tlb"
using namespace XMLSpyLib;

HRESULT CXMLSpyIDEPlugIn::OnCommand(long nID, IDispatch* pXMLSpy)
{
   USES_CONVERSION;

   if(nID == 1)     {
      IApplicationPtr      ipSpyApp;

      if(pXMLSpy)   {
         if(SUCCEEDED(pXMLSpy->QueryInterface(__uuidof(IApplication),(void **)
&ipSpyApp))) {
            IDocumentPtr  ipDocPtr = ipSpyApp->ActiveDocument;

            // we assume that grid view is active
            if(ipDocPtr) {
               IGridViewPtr        ipGridPtr = ipDocPtr->GridView;

               if(ipGridPtr)      {
                  IXMLDataPtr    ipXMLData = ipGridPtr->CurrentFocus;

                  CString  strValue = W2T(ipXMLData->TextValue);

                  if(!strValue.IsEmpty() && (strValue.Left(7) == _T("http://")))
                     ::ShellExecute(NULL,_T("open"),W2T(ipXMLData-
>TextValue),NULL,NULL,SW_SHOWNORMAL);
               }
            }
         }
      }
   }

   return S_OK;
}


HRESULT CXMLSpyIDEPlugIn::OnUpdateCommand(long nID, IDispatch* pXMLSpy, SPYUpdateAction*
pAction)
{
   *pAction = spyDisable;

   if(nID == 1)     {
      IApplicationPtr      ipSpyApp;
```

```
      if(pXMLSpy)   {
         if(SUCCEEDED(pXMLSpy->QueryInterface(__uuidof(IApplication),(void **)
&ipSpyApp))) {
            IDocumentPtr   ipDocPtr = ipSpyApp->ActiveDocument;

            // only enable if grid view is active
            if((ipDocPtr != NULL) && (ipDocPtr->CurrentViewMode == spyViewGrid))
               *pAction = spyEnable;
         }
      }
   }

   return S_OK;
}


HRESULT CXMLSpyIDEPlugIn::OnEvent(long nEventID, SAFEARRAY **arrayParameters, IDispatch*
pXMLSpy, VARIANT* pReturnValue)
{
   return S_OK;
}

HRESULT CXMLSpyIDEPlugIn::GetUIModifications(BSTR* pModificationsXML)
{
   CComBSTR  bstrMods = _T(" \
            <ConfigurationData>   \
               <Modifications>    ");
   // add "Open URL..." to Tools menu
   bstrMods.Append (_T(" \
                  <Modification> \
                     <Action>Add</Action> \
                     <UIElement type=\"MenuItem\"> \
                        <ID>1</ID> \
                        <Name>Open URL...</Name> \
                        <Place>0</Place> \
                        <MenuID>129</MenuID> \
                        <Parent>:Tools</Parent> \
                     </UIElement> \
                  </Modification>"));
   // add Seperator to Tools menu
   bstrMods.Append (_T(" \
                  <Modification> \
                     <Action>Add</Action> \
                     <UIElement type=\"MenuItem\"> \
                        <ID>0</ID> \
                        <Place>1</Place> \
                        <MenuID>129</MenuID> \
                        <Parent>:Tools</Parent> \
                     </UIElement> \
                  </Modification>"));
   // finish modification description
   bstrMods.Append (_T(" \
               </Modifications> \
            </ConfigurationData>"));

   return bstrMods.CopyTo(pModificationsXML);
}


HRESULT CXMLSpyIDEPlugIn::GetDescription(BSTR* pDescription)
{
```

```
   CComBSTR  bstrDescr = _T("ATL C++ XMLSpy IDE PlugIn;This PlugIn demonstrates the
implementation of a simple ATL DLL as a IDE PlugIn for XMLSpy.");
   return bstrDescr.CopyTo(pDescription);
}
```

25.2.5 IXMLSpyPlugIn

Methods

OnCommand 1090
OnUpdateCommand 1091
OnEvent 1092
GetUIModifications 1093
GetDescription 1094

Description

If a DLL is added to XMLSpy as an IDE plug-in, it is necessary that it registers a COM component that answers to an `IXMLSpyPlugIn` interface with the reserved `uuid(88F2A622-4B7E-42CD-8D04-3C0E5389DD85)`. This is required for it to be recognized as a plug-in.

25.2.5.1 OnCommand

Declaration

```
OnCommand(nID as long, pXMLSpy as IDispatch)
```

Description

The `OnCommand()` method of the interface implementation is called each time a command added by the the IDE plug-in (menu item or toolbar button) is processed. `nID` stores the command ID defined by the `ID` element of the respective `UIElement`. `pXMLSpy` holds a reference to the dispatch interface of the `Application` object of XMLSpy.

Example

```
Public Sub IXMLSpyPlugIn_OnCommand(ByVal nID As Long, ByVal pXMLSpy As Object)
   If (Not (pXMLSpy Is Nothing)) Then
      Dim objDlg
      Dim objDoc As XMLSpyLib.Document
      Dim objSpy As XMLSpyLib.Application
      Set objSpy = pXMLSpy

      If nID = 3 Or nID = 5 Then
         Set objDlg = CreateObject("MSComDlg.CommonDialog")
         objDlg.Filter = "XML Files (*.xml)|*.xml|All Files (*.*)|*.*||"
         objDlg.FilterIndex = 1
         objDlg.ShowOpen

         If Len(objDlg.FileName) > 0 Then
            Set objDoc = objSpy.Documents.OpenFile(objDlg.FileName, False)
            Set objDoc = Nothing
         End If
      End If

      If nID = 4 Or nID = 6 Then
         Set objDlg = CreateObject("MSComDlg.CommonDialog")
         objDlg.Filter = "All Files (*.*)|*.*||"
```

```
         objDlg.Flags = cdlOFNPathMustExist
         objDlg.ShowSave

         If Len(objDlg.FileName) > 0 Then
            Set objDoc = objSpy.ActiveDocument

            If Not (objDoc Is Nothing) Then
               objDoc.SetPathName objDlg.FileName
               objDoc.Save
               Set objDoc = Nothing
            End If
         End If
      End If

      Set objSpy = Nothing
   End If
End Sub
```

25.2.5.2 OnUpdateCommand

Declaration

```
OnUpdateCommand(nID as long, pXMLSpy as IDispatch) as SPYUpdateAction
```

Description

The `OnUpdateCommand()` method is called each time the visible state of a button or menu item needs to be set. `nID` stores the command ID defined by the `ID` element of the respective `UIElement`. `pXMLSpy` holds a reference to the dispatch interface of the `Application` object.

Possible return values to set the update state are:

```
spyEnable    = 1
spyDisable   = 2
spyCheck     = 4
spyUncheck   = 8
```

Example

```
Public Function IXMLSpyPlugIn_OnUpdateCommand(ByVal nID As Long, ByVal pXMLSpy As Object)
As SPYUpdateAction
   IXMLSpyPlugIn_OnUpdateCommand = spyDisable

   If (Not (pXMLSpy Is Nothing)) Then
      Dim objSpy As XMLSpyLib.Application
      Set objSpy = pXMLSpy

      If nID = 3 Or nID = 5 Then
         IXMLSpyPlugIn_OnUpdateCommand = spyEnable
      End If
      If nID = 4 Or nID = 6 Then
         If objSpy.Documents.Count > 0 Then
            IXMLSpyPlugIn_OnUpdateCommand = spyEnable
         Else
            IXMLSpyPlugIn_OnUpdateCommand = spyDisable
         End If
      End If
   End If
End Function
```

25.2.5.3 OnEvent

Declaration

```
OnEvent(nEventID as long, arrayParameters as SAFEARRAY(VARIANT), pXMLSpy as IDispatch) as
VARIANT
```

Description

`OnEvent()` is called each time an event is raised from XMLSpy.

Possible values for `nEventID` are:

```
On_BeforeStartEditing              = 1
On_EditingFinished                 = 2
On_FocusChanged                    = 3
On_Beforedrag                      = 4
On_BeforeDrop                      = 5
On_OpenProject                     = 6
On_OpenDocument                    = 7
On_CloseDocument                   = 8
On_SaveDocument                    = 9
On_DocEditDragOver                 = 10
On_DocEditDrop                     = 11
On_DocEditKeyDown                  = 12
On_DocEditKeyUp                    = 13
On_DocEditKeyPressed               = 14
On_DocEditMouseMove                = 15
On_DocEditButtonUp                 = 16
On_DocEditButtonDown               = 17
On_DocEditContextMenu              = 18
On_DocEditPaste                    = 19
On_DocEditCut                      = 20
On_DocEditCopy                     = 21
On_DocEditClear                    = 22
On_DocEditSelectionChanged         = 23
On_DocEditDragOver                 = 10
On_BeforeOpenProject               = 25
On_BeforeOpenDocument              = 26
On_BeforeSaveDocument              = 27
On_BeforeCloseDocument             = 28
On_ViewActivation                  = 29
On_DocEditKeyboardEvent            = 30
```


(C) 2015-2021 Altova GmbH

```
On_DocEditMouseEvent                = 31
On_BeforeValidate                   = 32
On_BeforeShowSuggestions            = 33
On_ProjectOpened                    = 34
On_Char                             = 35
On_Initialize                       = 36
On_Running                          = 37
On_Shutdown                         = 38
On_AuthenticBeforeSave              = 39
On_AuthenticContextMenuActivated    = 40
On_AuthenticLoad                    = 41
On_AuthenticToolbarButtonClicked    = 42
On_AuthenticToolbarButtonExecuted   = 43
On_AuthenticUserAddedXMLNode        = 44
```

The names of the events are the same as they appear in the Scripting Environment of XMLSpy. For IDE plug-ins the names used are immaterial. The events are identified using the ID value.

`arrayParameters` is an array which is filled with the parameters of the currently raised event. Order, type, and meaning of the single parameters are available through the scripting environment of XMLSpy. The events module of a scripting project contains predefined functions for all events prior to version 4.4. The parameters passed to the predefined functions are identical to the array elements of the `arrayParameters` parameter.

Events raised from the Authentic View of XMLSpy do not pass any parameters directly. An "event" object is used instead. The event object can be accessed through the `Document` object of the active document.

`pXMLSpy` holds a reference to the dispatch interface of the `Application` object of XMLSpy.

If the return value of `OnEvent()` is set, then neither the IDE plug-in nor an event handler inside of the scripting environment will get this event afterwards. Please note that all IDE plug-ins get/process the event before the Scripting Environment does.

25.2.5.4 GetUIModifications

Declaration

```
GetUIModifications() as String
```

Description

The `GetUIModifications()` method is called during initialization of the plug-in, to get the configuration XML data that defines the changes to the UI of XMLSpy. The method is called when the plug-in is loaded for the first time, and at every start of XMLSpy. *See also Configuration XML* 1082 *for a detailed description how to change the UI.*

Example

```
Public Function IXMLSpyPlugIn_GetUIModifications() As String
   ' GetUIModifications() gets the XML file with the specified modifications of
   ' the UI from the config.xml file in the plug-in folder
```

```
   Dim strPath As String
   strPath = App.Path

   If Len(strPath) > 0 Then
      Dim fso As New FileSystemObject
      Dim file As file
      Set file = fso.GetFile(strPath & "\config.xml")

      If (Not (file Is Nothing)) Then
         Dim stream As TextStream
         Set stream = file.OpenAsTextStream(ForReading)

         ' this replaces the token '**path**' from the XML file with
         ' the actual installation path of the plug-in to get the image file
         Dim strMods As String
         strMods = stream.ReadAll
         strMods = Replace(strMods, "**path**", strPath)

         IXMLSpyPlugIn_GetUIModifications = strMods
      Else
         IXMLSpyPlugIn_GetUIModifications = ""
      End If
   End If
End Function
```

25.2.5.5 GetDescription

Declaration

```
GetDescription() as String
```

Description

`GetDescription()` is used to define the description string for the plug-in entries visible in the Customize dialog box.

Example

```
Public Function IXMLSpyPlugIn_GetDescription() As String
    IXMLSpyPlugIn_GetDescription = "Sample Plug-in for XMLSpy;This Plug-in demonstrates
the implementation of a simple VisualBasic DLL as a Plug-in for XMLSpy."
End Function
```


(C) 2015-2021 Altova GmbH

25.3 Application API

The COM-based API of XMLSpy (also called the Application API from now on) enables other applications to use the functionality of XMLSpy. As a result, it is possible to automate a wide range of tasks, from validating an XML file to modifying complex XML content (with the XMLData[1309] interface).

XMLSpy and its Application API follow the common specifications for automation servers set out by Microsoft. It is possible to access the methods and properties of the Application API from common development environments, such as those using C#, C++, VisualBasic, and Delphi, and with scripting languages like JScript and VBScript.

Execution environments for the Application API

The Application API can be accessed from the following execution environments:

- External programs (described below[1095] and in the Overview[1096] part of this section)
- From within the built-in Scripting Editor of XMLSpy. For a description of the scripting environment, see the section, Scripting Editor[1056].
- XMLSpy allows you to create and integrate your own plug-ins into the application using a special interface for plug-ins. A description of how to create plug-ins is given in the section IDE Plug-ins[1081].
- Via an ActiveX Control, which is available if the integration package[1413] is installed. For more information, see the section ActiveX Integration[1413].

External programs

In the Overview[1096] part of this section, we describe how the functionality of XMLSpy can be accessed and automated from external programs.

Using the Application API from outside XMLSpy requires an instance of XMLSpy to be started first. How this is done depends on the programming language used. See the section, Programming Languages[1097], for information about individual languages.

Essentially, XMLSpy will be started via its COM registration. Then the `Application` object associated with the XMLSpy instance is returned. Depending on the COM settings, an object associated with an already running XMLSpy can be returned. Any programming language that supports creation and invocation of COM objects can be used. The most common of these are listed below.

- JScript and VBScript[1106] script files have a simple syntax and are designed to access COM objects. They can be run directly from a DOS command line or with a double click on Windows Explorer. They are best used for simple automation tasks.
- C#[1109] is a full-fledged programming language that has a wide range of existing functionality. Access to COM objects can be automatically wrapped using C#..
- C++ provides direct control over COM access but requires relatively larger amounts of code than the other languages.
- Java[1118]: Altova products come with native Java classes that wrap the Application API and provide a full Java look-and-feel.
- Other programming languages that make useful alternatives are: Visual Basic for Applications, Perl, and Python.

Programming points

The following limitations must be considered in your client code:

- Be aware that if your client code crashes, instances of XMLSpy may still remain in the system.
- Don't hold references to objects in memory longer than you need them, especially those from the `XMLData` interface. If the user interacts between two calls of your client, then there is no guarantee that these references are still valid.
- Don't forget to disable dialogs if the user interface is not visible.
- See Error handling in JScript[1101] (and in C#[1116] and Java[1125]) for details of how to avoid annoying error messages.
- Free references explicitly if you are using C# or C++.

This documentation

This documentation section about the Application API is broadly divided into two parts.

- The first part consists of an Overview[1096], which describes the object model for the API and explains how the API is accessed via various programming languages[1097].
- The second part is a reference section (Interfaces[1130] and Enumerations[1358]) that contains descriptions of the interface objects of the Application API.

25.3.1 Overview

This overview of the Application API is organized as follows:

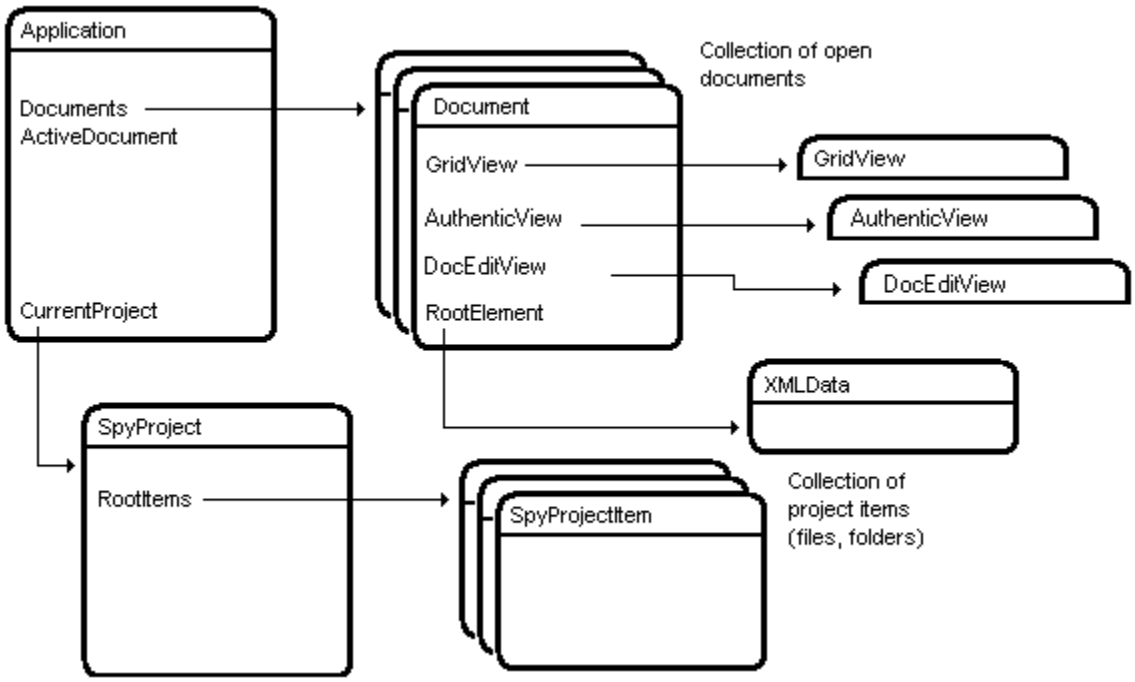
- The Object Model[1096] describes the relationships between the objects of the Application API.
- Programming Languages[1097] explains how the most commonly used programming languages (JScript, VBScript, C#, and Java) can be used to access the functionality of the Application API. Code listings from the example files supplied with your application package are used to describe basic mechanisms.
- The DOM and XMLData[1126] explains the relationship between the Application API's XMLData interface and the DOM.
- Obsolete: Authentic View Row Operations[1128] supplies information about obsolete objects for Authentic View table row operations.
- Obsolete: Authentic View Editing Operations[1129] supplies information about obsolete objects for Authentic View editing operations.

25.3.1.1 Object Model

The starting point for every application which uses the Application API is the `Application`[1131] object. This object contains general methods like import/export support and references to the open documents and any open project.

The `Application` object is created differently in various programming languages. In scripting languages such as JScript or VBScript, this involves calling a function which initializes the application's COM object. For examples, see the Programming Languages[1097] section.

The picture below shows the links between the main objects of the Application API:



The application object consists of the following parts:

1. Document collection and reference to the active document.
2. Reference to current project and methods for creating and opening projects.
3. Methods to support the export to and import from databases, text files, and Word documents.
4. URL management.
5. Methods for macro menu items.

Once you have created an Application object you can start using the functionality of XMLSpy. In most cases, you either open a project and access the documents from there or you directly open a document via the Documents 1246 interface.

25.3.1.2 Programming Languages

Programming languages differ in the way they support COM access. A few examples for the most frequently used languages (*links below*) will help you get started. The code listings in this section show how basic functionality can be accessed. The files in the `API` subfolder of the `Examples` folder can be used to test this functionality:

Windows 7、Windows 8、Windows 10	`C:\Users\<username>\Documents\Altova\XMLSpy\2021\Examples`

JScript

The JScript listings demonstrate the following basic functionality:

- Start application or attach to a running instance 1099

- Simple document access 1100
- Iteration 1101
- Error handling 1101
- Events 1102
- Import and export of data 1103

VBScript

VBScript is different than JScript only syntactically; otherwise it works in the same way. The link below goes to an example of how VBScript can be used. For more information, refer to the JScript examples 1098.

- Events 1107: Shows how events are handled using VBScript.

C#

C# can be used to access the Application API functionality. The code listings show how to access the API for certain basic functionality.

- Start XMLSpy 1114: Starts XMLSpy, which is registered as an automation server, or activates the program if XMLSpy is already running.
- Open OrgChart.pxf 1115: Locates one of the example documents installed with XMLSpy and opens it. If this document is already open it becomes the active document.
- OnDocumentOpened Event On/Off 1117: Shows how to listen to XMLSpy events. When turned on, a message box will pop up after a document has been opened.
- Open ExpReport.xml 1115: Opens another example document.
- Toggle View Mode 1116: Changes the view of all open documents between Text View and Authentic View. The code shows how to iterate through open documents.
- Validate 1116: Validates the active document and shows the result in a message box. The code shows how to handle errors and COM output parameters.
- Shutdown XMLSpy 1114: Stops XMLSpy.

Java

The XMLSpy API can be accessed from Java code. The Java sub-section of this section 1118 explains how some basic XMLSpy functionality can be accessed from Java code. It is organized into the following sub-sections:

- Mapping Rules for the Java Wrapper 1118
- Example Java Project 1120
- Application Startup and Shutdown 1123
- Simple Document Access 1124
- Iterations 1124
- Use of Out-Parameters 1125
- Event Handlers 1125

25.3.1.2.1 JScript

This section contains listings of JScript code that demonstrate the following basic functionality:

- Start application or attach to a running instance 1099

- Simple document access 1100
- Iteration 1101
- Error handling 1101
- Events 1102
- Import and export of data 1103

Example files

The code listings in this section are available in example files that you can test as is or modify to suit your needs. The JScript example files are located in the `JScript` subfolder of the API Examples folder:

Windows 7、Windows 8、Windows 10	`C:\Users\<username>\Documents\Altova\XMLSpy\2021\Examples`

The example files can be run in one of two ways:

- *From the command line:* Open a command prompt window, change the directory to the path above, and type the name of one of the example scripts (for example, `Start.js`).
- *From Windows Explorer:* In Windows Explorer, browse for the JScript file and double-click it.

The script is executed by Windows Script Host that is packaged with Windows operating system. For more information about Windows Script Host, refer to MSDN documentation (https://msdn.microsoft.com).

25.3.1.2.1.1 Start Application

The JScript below starts the application and shuts it down. If the COM object of the 32-bit XMLSpy cannot be found, the code attempts to get the COM object of the 64-bit application; otherwise, an error is thrown. If an instance of the application is already running, the running instance will be returned.

メモ 32-ビット XMLSpy では、登録された名前、またはCOM オブジェクトのプログラム識別子（ProgId）は以下の通りです: `MapForce.ApplicationXMLSpy.Application`。64-ビット XMLSpy では、名前は以下の通りです: `MapForce_x64.ApplicationXMLSpy_x64.Application`。プログラムの呼び出しは自身のレジストリハイブ、またはグループ（32-bit または64-bit）内のCLASSES レジストリエントリにアクセスすることに注意してください。ですから、標準のコマンドプロンプトと上 64-bit Windows のWindows Explorerを使用してスクリプトを実行すると、64-ビット XMLSpy. を指す64-ビットレジストリエントリにアクセスされます。この理由のため、XMLSpy 32-ビットと64-ビットの両方がインストールされている場合、32-ビット XMLSpy. を呼び出すために特別な処理が必要になります。例えば、スクリプトホストがプログラムを呼び出す場合以下を行います:

1. ディレクトリを **C:\Windows\SysWOW64** に変更します。
2. コマンドライン上で **wscript.exe** を入力し実行するスクリプトパスを以下のように入力します:

```
wscript.exe "C:\Users\...
\Documents\Altova\StyleVision2021\StyleVisionExamples\API\JScript\start.js"wscript.exe
"C:\Users\...\Documents\Altova\XMLSpy2021\Examples\API\JScript\start.js"
```

```
// Initialize application's COM object. This will start a new instance of the
application and
```

```
// return its main COM object. Depending on COM settings, the main COM object of an
already
// running application might be returned.

try {   objSpy = WScript.GetObject("", "XMLSpy.Application");   }
catch(err) {}

if( typeof( objSpy ) == "undefined" )
{
   try   {   objSpy = WScript.GetObject("", "XMLSpy_x64.Application")   }
   catch(err)
   {
      WScript.Echo( "Can't access or create XMLSpy.Application" );
      WScript.Quit();
   }
}

// if newly started, the application will start without its UI visible. Set it to
visible.
objSpy.Visible = true;

WScript.Echo(objSpy.Edition + " has successfully started. ");

objSpy.Visible = false;      // will shutdown application if it has no more COM
connections
//objSpy.Visible = true;   // will keep application running with UI visible
```

The JScript code listed above is available in the sample file `Start.js` (see Example Files[1099]).

25.3.1.2.1.2 Simple Document Access

After you have started the application as shown in Start Application[1099], you will most likely want to programmatically open a document in order to work with it. The JScript code listing below illustrates how to open two documents from the XMLSpy Examples folder and set one of them as the active document.

```
// Locate examples via USERPROFILE shell variable. The path needs to be adapted to
major release versions.
objWshShell = WScript.CreateObject("WScript.Shell");
majorVersionYear = objSpy.MajorVersion + 1998
strExampleFolder = objWshShell.ExpandEnvironmentStrings("%USERPROFILE%") + "\\My
Documents\\Altova\\XMLSpy" + majorVersionYear + "\\Examples\\";

// Tell XMLSpy to open two documents. No dialogs
objDoc1 = objSpy.Documents.OpenFile(strExampleFolder + "OrgChart.pxf", false);
objSpy.Documents.OpenFile(strExampleFolder + "ExpReport.xml", false);

// The document currently active can be easily located.
objDoc2 = objSpy.ActiveDocument;

// Let us make sure that the document is shown in grid view.
objDoc2.SwitchViewMode(0);   // SPYViewModes.spyViewGrid = 0

// Now switch back to the document opened first
objDoc1.SetActiveDocument();
```

The JScript code listed above is available in the sample file `DocumentAccess.js` (see Example Files 1099).

25.3.1.2.1.3 Iteration

The JScript listing below shows how to iterate through the open documents. It is assumed that you have already started the application and opened some documents as shown in the previous sections.

```
// go through all open documents using a JScript Enumerator
bRequiresSaving = false;
for (var iterDocs = new Enumerator(objSpy.Documents); !iterDocs.atEnd();
iterDocs.moveNext())
{
   if (iterDocs.item().IsModified)
      bRequiresSaving = true;

   var strErrorText = new Array(1);
   var nErrorNumber = new Array(1);
   var errorData = new Array(1);

   if (!iterDocs.item().IsValid(strErrorText, nErrorNumber, errorData))
   {
       var text = strErrorText;
      // access that XMLData object only if filled in
       if (errorData[0] != null)
         text += "(" + errorData[0].Name + "/" + errorData[0].TextValue + ")";

      WScript.Echo("Document \"" + iterDocs.item().Name +"\" validation error[" +
nErrorNumber + "]: " + text);
   }
   else
   {
      // The COM call succeeded and the document is valid.
      WScript.Echo("Document \"" + iterDocs.item().Name + "\" is valid.");
   }
}

// go through all open documents using index-based access to the document collection
for (i = objSpy.Documents.Count; i > 0; i--)
   objSpy.Documents.Item(i).Close(false);
```

The JScript code listed above is available in the sample file `DocumentAccess.js` (see Example Files 1099).

25.3.1.2.1.4 Error Handling

The Application API returns errors in two different ways:

- The `HRESULT` returned by every API method
- The `IErrorInfo` interface of the Application API

Every API method returns an HRESULT. This return value gives the caller information about errors during execution of the method. If the call was successful, the return value is S_OK. The HRESULT option is commonly used in C/C++ programs.

However, programming languages such as VisualBasic and scripting languages (and other high-level development environments) don't give the programmer access to the HRESULT return of a COM call. Such languages use the IErrorInfo interface, which is also supported by the Application API. If an error occurs, the Application API creates a new object that implements the IErrorInfo interface. The information provided by the IErrorInfo interface is imported by the development environment into its own error-handling mechanism.

For example, the JScript code listing below causes an error to be thrown by incorrectly declaring an array. Additional information about the error object is provided by its properties number and description.

```
try {
   var arr = new Array(-1);
}
catch (err) {
   WScript.Echo("Error : (" + (err.number & 0xffff) + ")" + err.description);
}
```

25.3.1.2.1.5 Events

COM specifies that a client must register itself at a server for callbacks using the connection point mechanism. The automation interface for XMLSpy defines the necessary event interfaces. The way to connect to those events depends on the programming language you use in your client. The following code listing shows how this is done using JScript.

The method WScript.ConnectObject is used to receive events.

```
// The event-handler function
function DocEvent_OnBeforeCloseDocument(objDocument)
{
   WScript.Echo("Received event - before closing document");
}

// Create or connect to XMLSpy (or Authentic Desktop)
try
{
   // Create the environment and XMLSpy (or Authentic Desktop)
   objWshShell = WScript.CreateObject("WScript.Shell");
   objFSO = WScript.CreateObject("Scripting.FileSystemObject");
   objSpy = WScript.GetObject("", "XMLSpy.Application");

// If only Authentic Desktop is installed (and XMLSpy is not installed) use:
// objSpy = WScript.GetObject("", "AuthenticDesktop.Application")

}
catch(err)
   { WScript.Echo ("Can't create WScript.Shell object or XMLSpy"); }

// Create document object and connect to its events
objSpy.Visible = true;
majorVersionYear = objSpy.MajorVersion + 1998
```

```
docPath = objWshShell.ExpandEnvironmentStrings("%USERPROFILE%") + "\\Documents\\Altova\
\XMLSpy" + majorVersionYear + "\\Examples\\ExpReport.xml";
objDoc = objSpy.Documents.OpenFile (docPath, false);
WScript.ConnectObject(objDoc, "DocEvent_");

// Keep running while waiting for the event
// In the meanwhile close this document in XMLSpy (or Authentic Desktop) manually
WScript.Echo ("Sleeping for 10 seconds ...");
WScript.Sleep (10000);

objDoc = null;
WScript.Echo ("Stopped listening for event");
objSpy.Quit();
```

25.3.1.2.1.6 Import and Export of Data

Before you implement your import and export tasks with the Application API, it is good practice to test the connections, parameters, SQL queries and so on in XMLSpy. In this way you are able to verify the results and make quick adjustments to import or export parameters. Most of the methods for importing and exporting data are placed in the `Application` 1131 object; the remaining functions are accessible via the `Document` 1218 interface.

There is some preparatory work necessary before the actual import or export can be started. Every import/export job consists of two parts. You need to define a connection to your data and the specific behavior for the import/export process. In case of an import, the connection is either a database, a text-file, or a Word document. The main issue is which data (columns) to imported into XMLSpy. In case of an export, the connection is either a database or a text file. Specify which data (elements of the XML file) and additional parameters (for example, automatic key generation or number of sub-levels) to use from the XML-structure.

The properties in the `DatabaseConnection`, `TextImportExportSettings` and `ExportSettings` interfaces have default values. See the corresponding descriptions in the `Interfaces` 1130 chapter for further information.

The sub-sections of this section describe each of these operations in detail.

- Import from Database 1103
- Export to Database 1104
- Import from Text 1105
- Export to Text 1106

Given below are the steps to establish a connection to an existing database for import:

1. Use a `DatabaseConnection` 1209 object and set the following:
   - The method `Application.GetDatabaseSettings` 1138 returns a new object for a database connection: `objImpSettings = objSpy.GetDatabaseSettings();`
   - You have to set either an ADO connection string, `objImpSettings.ADOConnection = strADOConnection` or the path to an existing database file: `objImpSettings.File = strExampleFolder + "Tutorial\\Company.mdb";`
   - To complete the settings you create an SQL SELECT statement to define the data to be queried: `objImpSettings.SQLSelect = "SELECT * FROM Address";`
2. Call `Application.GetDatabaseImportElementList` 1137 to get a collection of the resulting columns of the SQL query: `objElementList = objSpy.GetDatabaseImportElementList(objImpSettings);` This collection gives you the

opportunity to control which columns should be imported and specify the datatype of the new elements. Each item of the collection represents one column to import. If you remove an item, the corresponding column will not be imported. You can additionally modify the ElementListItem.ElementKind[1256] property to set the datatype of the XML elements for each column. Please consider that GetDatabaseImportElementList() executes the SQL query and could initiate a time-consuming call. To avoid this, it is possible to pass a null-pointer as the second parameter to ImportFromDatabase(); this imports all columns as plain XML elements.

3. Start the import with Application.ImportFromDatabase[1141]: objImpDocFromDB = objSpy.ImportFromDatabase(objImpSettings,objElementList);

```
// Locate examples via USERPROFILE shell variable.
objWshShell = WScript.CreateObject("WScript.Shell");
majorVersionYear = objSpy.MajorVersion + 1998
strExampleFolder = objWshShell.ExpandEnvironmentStrings("%USERPROFILE%") + "\\My
Documents\\Altova\\XMLSpy" + majorVersionYear + "\\Examples\\";

try
{
   // specify the source of data import
   objImpSettings = objSpy.GetDatabaseSettings();
   objImpSettings.File = strExampleFolder + "Tutorial\\Company.mdb";
   objImpSettings.SQLSelect = "SELECT * FROM Address";

   // column filter
   objElementList = objSpy.GetDatabaseImportElementList(objImpSettings);

   // import into a new XML file
   objImpDocFromDB = objSpy.ImportFromDatabase(objImpSettings,objElementList);
}
catch(err)
{
   WScript.Echo("Error importing from database.\n\n" +
                     "Error: " + (err.number & 0xffff) + "\n" +
                     "Description: " + err.description);
}
```

The JScript code listed above is available in the sample file ImportExport.js (see Example Files[1099]).

To export data to a database, carry out the steps below:

1. Use a DatabaseConnection[1209] object and set the necessary properties. All properties except SQLSelect are important for the export. ADOConnection or File defines the target for the output. You need to set only one of them.
2. Fill an ExportSettings[1257] object with the required values. These properties are the same options as those available in the export dialog of XMLSpy. Select the menu option **Convert | Export to Text files/Database** to see the options and try a combination of export settings. After that it is easy to transfer these settings to the properties of the interface. Call Application.GetExportSettings[1139] to get an ExportSettings object: objExpSettings = objSpy.GetExportSettings()
3. Build an element list with Document.GetExportElementList[1235]. The element list enables you to eliminate XML elements from the export process. It also gives you information about the record and field count in the RecordCount and FieldCount properties. Set the ExportSettings.ElementList[1258] property to this collection. It is possible to set the element list to null/Nothing (default) to export all elements.
4. Call Document.ExportToDatabase[1230] to execute the export. The description of the ExportToDatabase method contains also a code example for a database export.

```
   // set the behaviour of the export with ExportSettings
   objExpSettings = objSpy.GetExportSettings()

   //set the destination with DatabaseConnection
   objDB = objSpy.GetDatabaseSettings();
   objDB.CreateMissingTables = true;
   objDB.CreateNew = true;
   objDB.File = "C:\\Temp\\Export.mdb";

try
{
   objImpDocFromDB.ExportToDatabase(objImpDocFromDB.RootElement, objExpSettings,
objDB);
}
catch(err)
{
   WScript.Echo("Error exporting to database.\n\n" +
                    "Error: " + (err.number & 0xffff) + "\n" +
                    "Description: " + err.description);
}
```

The JScript code listed above is available in the sample file `ImportExport.js` (see Example Files[1099]).

Importing data from a text file is similar to the import from a database. You must use other interfaces (described in steps 1 to 3 below) with different methods and properties:

1. Use a `TextImportExportSettings`[1301] object and set the properties: The method `Application.GetTextImportExportSettings`[1140] returns a new object to specify a text file for import: `objImpSettings = objSpy.GetTextImportExportSettings();` You have to set at least the `ImportFile` property to the path of the file for the import. Another important property is `HeaderRow`. Set it to `False` if the text file does not contain a leading line as a header row: `objImpSettings.ImportFile = strExampleFolder + "Tutorial\\Shapes.txt";`
2. Call `Application.GetTextImportElementList`[1139] to get a collection of all columns inside the text file: `objElementList = objSpy.GetTextImportElementList(objImpSettings);`
3. Start the import with `Application.ImportFromText`[1143]: `objImpDocFromText = objSpy.ImportFromText(objImpSettings,objElementList);`

```
try
{
   // specify the source of data import
   objImpSettings = objSpy.GetTextImportExportSettings();
   objImpSettings.ImportFile = strExampleFolder + "Tutorial\\Shapes.txt";
   objImpSettings.HeaderRow = false;

   // column filter
   objElementList = objSpy.GetTextImportElementList(objImpSettings);

   // import into a new XML file
   objImpDocFromText = objSpy.ImportFromText(objImpSettings,objElementList);
}
catch(err)
{
   WScript.Echo("Error importing from text file.\n\n" +
                    "Error: " + (err.number & 0xffff) + "\n" +
```

```
                    "Description: " + err.description);
}
```

The JScript code listed above is available in the sample file `ImportExport.js` (see Example Files[1099]).

To export data to text, carry out the steps below:

1. Use a `TextImportExportSettings`[1301] object and set the necessary properties.
2. Fill an `ExportSettings`[1257] object with the required values. See Item 2 in Export to database[1104].
3. Build an element list with `Document.GetExportElementList`[1235]. See Item 3 in Export to database[1104].
4. Call `Document.ExportToText`[1231] to execute the export.

```
objExpSettings = objSpy.GetExportSettings();
objExpSettings.ElementList =
objImpDocFromText.GetExportElementList(objImpDocFromText.RootElement, objExpSettings);

objTextExp = objSpy.GetTextImportExportSettings();
objTextExp.HeaderRow = true;
objTextExp.DestinationFolder = "C:\\Temp";

try
{

       objImpDocFromText.ExportToText(objImpDocFromText.RootElement, objExpSettings,
objTextExp);
}
catch(err)
{
   WScript.Echo("Error exporting to text.\n\n" +
                    "Error: " + (err.number & 0xffff) + "\n" +
                    "Description: " + err.description);
}
```

The JScript code listed above is available in the sample file `ImportExport.js` (see Example Files[1099]).

25.3.1.2.2 VBScript

VBScript is syntactically different than JScript but works in the same way. This section contains a listing showing how events are used with VBScript[1107] and an example[1108].

For information about other functionality, refer to the JScript examples listed below:

- Start application or attach to a running instance[1099]
- Simple document access[1100]
- Iteration[1101]
- Error handling[1101]
- Import and export of data[1103]

(C) 2015-2021 Altova GmbH

25.3.1.2.2.1 Events

COM specifies that a client must register itself at a server for callbacks using the connection point mechanism. The automation interface for XMLSpy defines the necessary event interfaces. The way to connect to those events depends on the programming language you use in your client. The following code listing shows how this is done using VBScript.

The method `WScript.ConnectObject` is used to receive events.

To run this code, paste it into a file with .vbs extension, and either double-click in Windows Explorer, or run it from a command prompt.

```
' the event handler function
Function DocEvent_OnBeforeCloseDocument(objDocument)
   Call WScript.Echo("received event - before closing document")
End Function

' create or connect to XmlSpy
Set objWshShell = WScript.CreateObject("WScript.Shell")
Set objFSO = WScript.CreateObject("Scripting.FileSystemObject")
Set objSpy = WScript.GetObject("", "XMLSpy.Application")
' If only Authentic is installed (and XMLSpy is not installed) use:
' Set objSpy = WScript.GetObject("", "AuthenticDesktop.Application")
' If only XMLSpy 64-bit is intalled, use:
' Set objSpy = WScript.GetObject("", "XMLSpy_x64.Application")

' create document object and connect to its events
objSpy.Visible = True

' Find out user's personal folder and locate one of the installed examples.
personalFolder = objWshShell.ExpandEnvironmentStrings("%UserProfile%")
majorVersionYear = objSpy.MajorVersion + 1998
xmlspyExamplesFolder = personalFolder & "\Documents\Altova\XMLSpy" & majorVersionYear
& "\Examples\"
docPath = xmlspyExamplesFolder & "ExpReport.xml"

' open a document
Set objDoc = objSpy.Documents.OpenFile (docPath, False)
Call WScript.ConnectObject(objDoc, "DocEvent_")

' keep running while waiting on the event
' in the meantime close the document in XMLSPY manually
Call WScript.Echo ("sleeping for 10 seconds ...")
Call WScript.Sleep (10000)

Set objDoc = Nothing
Call WScript.Echo ("stopped listening for event")
Call objSpy.Quit
```

メモ 32-ビット XMLSpy では、登録された名前、またはCOM オブジェクトのプログラム識別子（ProgId）は以下の通りです: `MapForce.ApplicationXMLSpy.Application`。64-ビット XMLSpy では、名前は以下の通りです: `MapForce_x64.ApplicationXMLSpy_x64.Application`。プログラムの呼び出しは自身のレジストリハイブ、またはグループ（32-bit または64-bit）内のCLASSES レジストリエントリにアクセスすることに注意してください。ですから、標準のコマンドプロンプトと上 64-bit Windows のWindows Explorerを使用してスクリプトを実行すると、64-ビット XMLSpy. を指

す 64-ビットレジストリエントリにアクセスされます。この理由のため、XMLSpy 32-ビットと64-ビットの両方がインストールされている場合、32-ビット XMLSpy. を呼び出すために特別な処理が必要になります。例えば、スクリプトホストがプログラムを呼び出す場合以下を行います:

1. ディレクトリを **C:\Windows\SysWOW64** に変更します。
2. コマンドライン上で **wscript.exe** を入力し実行するスクリプトパスを以下のように入力します:

```
wscript.exe "C:\Users\...
\Documents\Altova\StyleVision2021\StyleVisionExamples\API\JScript\start.js"wscript.exe
"C:\Users\...\Documents\Altova\XMLSpy2021\Examples\API\JScript\start.js"
```

25.3.1.2.2.2 Example: Using Events

Authentic View supports event connection on a per-object basis. Implementation of this feature is based on COM connection points and is available in environments that support this mechanism.

The following example is a VBScript code example that shows how to use events from within a VBScript project.

```
' ---------------------------------------------------------------------------
' VBScript example that demonstrates how to use events.
' ---------------------------------------------------------------------------

' Event handler for OnSelectionChanged event of AuthenticView
Function AuthenticViewEvent_OnSelectionChanged(objAuthenticRange)
    If objAuthenticRange.FirstTextPosition <> objAuthenticRange.LastTextPosition Then
        Call WScript.Echo("Selection: " & objAuthenticRange.Text & vbNewLine &
vbNewLine & "Close this dialog.")
    Else
        Call WScript.Echo("Cursor position: " & objAuthenticRange.FirstTextPosition &
vbNewLine & vbNewLine & "Close this dialog.")
    End If
End Function

' Start/access XMLSpy and connect to its automation interface.
Set WshShell = WScript.CreateObject("WScript.Shell")
Set objSpy = GetObject("", "XMLSpy.Application")
' Make the UI of XMLSpy visible.
objSpy.Visible = True

' Find out user's personal folder and locate one of the installed XMLSpy examples.
personalFolder = WshShell.ExpandEnvironmentStrings("%UserProfile%")
majorVersionYear = objSpy.MajorVersion + 1998
xmlspyExamplesFolder = personalFolder & "\Documents\Altova\XMLSpy" & majorVersionYear
& "\Examples\"
docPath = xmlspyExamplesFolder & "ExpReport.xml"

' Create object to access windows file system and test if the our document exists.
Set fso = CreateObject("Scripting.FileSystemObject")
If fso.FileExists(docPath) Then
    ' open the document
    Call objSpy.Documents.OpenFile(docPath, False)
    set objDoc = objSpy.ActiveDocument

    ' switch active document to authentic view
```

```
    objDoc.SwitchViewMode 4 ' spyViewAuthentic

    ' Register for connection point events on the authentic view of the active
document.
    ' Any function with a valid event name prefixed with "AuthenticViewEvent_" will
    ' be called when the corresponding event gets triggered on the specified object.
    set objView = objDoc.AuthenticView
    Call WScript.ConnectObject(objView, "AuthenticViewEvent_")
    Call WScript.Echo("Events are connected." & vbNewLine & vbNewLine & "Now set or
move the cursor in XMLSpy." & vbNewLine & vbNewLine & "Close this dialog to shut down
XMLSpy.")

    ' To disconnect from the events delete the reference to the object.
    set objView = Nothing
Else
    Call WScript.Echo("The file " & docPath & " does not exist.")
End If

' shut down XMLSpy when this script ends
objSpy.Visible = False
```

25.3.1.2.3 C#

The C# programming language can be used to access the Application API functionality. You could use Visual Studio 2010/2012/2013/2015/2017/2019 to create the C# code, saving it in a Visual Studio project. Create the project as follows:

1. In Microsoft Visual Studio, add a new project using **File | New | Project**.
2. Add a reference to the XMLSpy Type Library by clicking **Project | Add Reference**. The Add Reference dialog appears. Browse for the XMLSpy Type Library component, which is located in the XMLSpy application folder, and add it.
3. Enter the code you want.
4. Compile the code and run it.

Example C# project

Your XMLSpy package contains an example C# project, which is located in the `API\C#` subfolder of the `Examples` folder :

Windows 7、Windows 8、Windows 10	`C:\Users\<username>\Documents\Altova\XMLSpy\2021\Examples`

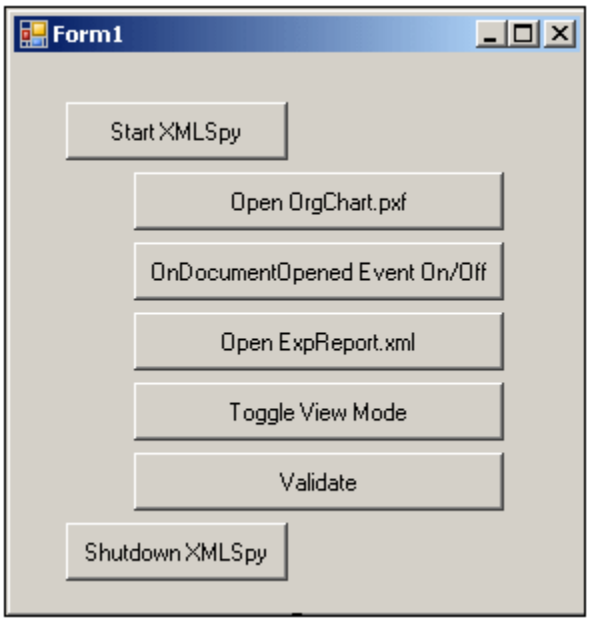
You can compile and run the project from within Visual Studio 2010/2012/2013/2015/2017/2019. The code listing below shows how basic application functionality can be used. This code is similar to the example C# project in the API Examples folder of your application package, but might differ slightly.

Platform configuration

If you have a 64-bit operating system and are using a 32-bit installation of XMLSpy, you must add the x86 platform in the solution's Configuration Manager and build the sample using this configuration. A new x86 platform (for the active solution in Visual Studio) can be created in the New Solution Platform dialog (**Build | Configuration Manager | Active solution platform | <New…>**).

What the code listing below does

The example code listing below creates a simple user interface (*screenshot below*) with buttons that invoke basic XMLSpy operations:



- Start XMLSpy 1114: Starts XMLSpy, which is registered as an automation server, or activates the application if it is already running.
- Open OrgChart.pxf 1115: Locates one of the example documents installed with XMLSpy and opens it. If this document is already open it becomes the active document.
- OnDocumentOpened Event On/Off 1117: Shows how to listen to XMLSpy events. When turned on, a message box will pop up after a document has been opened.
- Open ExpReport.xml 1115: Opens another example document.
- Toggle View Mode 1116: Changes the view of all open documents between Text View and Authentic View. The code shows how to iterate through open documents.
- Validate 1116: Validates the active document and shows the result in a message box. The code shows how to handle errors and COM output parameters.
- Shut down XMLSpy 1114: Stops XMLSpy.

You can modify the code (of the code listing below or of the example C# project in the API Examples folder) in any way you like and run it.

Compiling and running the example

In the API Examples folder, double-click the file `AutomateXMLSpy_VS2008.sln` or the file `AutomateXMLSpy_VS2010.sln` (to open in Visual Studio 2010/2012/2013/2015/2017/2019). Alternatively the file can be opened from within Visual Studio (with **File | Open | Project/Solution**). To compile and run the example, select **Debug | Start Debugging** or **Debug | Start Without Debugging**.

Code listing of the example

Given below is the C# code listing of the basic functionality of the form (`Form1.cs`) created in the `AutomateXMLSpy` example. Note that the code listed below might differ slightly from the code in the API

 (C) 2015-2021 Altova GmbH

Examples form.The listing below is commented for ease of understanding. Parts of the code are also presented separately in the sub-sections of this section, according to the Application API functionality they access.

The code essentially consists of a series of handlers for the buttons in the user interface shown in the screenshot above.

```
namespace WindowsFormsApplication2
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        // An instance of XMLSpy accessed via its automation interface
        XMLSpyLib.Application XMLSpy;

        // Location of examples installed with XMLSpy
        String strExamplesFolder;

        private void Form1_Load(object sender, EventArgs e)
        {
            // Locate examples installed with XMLSpy
            // REMARK: You might need to adapt this if you have a different major version
of the product
            strExamplesFolder = Environment.GetEnvironmentVariable("USERPROFILE") + "\\My
Documents\\Altova\\XMLSpy2012\\Examples\\";
        }

        // Handler for the "Start XMLSpy" button
        private void StartXMLSpy_Click(object sender, EventArgs e)
        {
            if (XMLSpy == null)
            {
                Cursor.Current = Cursors.WaitCursor;

                // If no XMLSpy instance is open, create one and make it visible
                XMLSpy = new XMLSpyLib.Application();
                XMLSpy.Visible = true;

                Cursor.Current = Cursors.Default;
            }
            else
            {
                // If an instance of XMLSpy is already running, make sure it's visible
                if (!XMLSpy.Visible)
                    XMLSpy.Visible = true;
            }
        }

        // Handler for the "Open OrgChart.pxf" button
        private void openOrgChart_Click(object sender, EventArgs e)
        {
            // Make sure there's a running XMLSpy instance, and that it's visible
            StartXMLSpy_Click(null, null);

            // Open one of the example files installed with the product
            XMLSpy.Documents.OpenFile(strExamplesFolder + "OrgChart.pxf", false);
```

```
        }

        // Handler for the "Open ExpReport.xml" button
        private void openExpReport_Click(object sender, EventArgs e)
        {
            // Make sure there's a running XMLSpy instance, and that it's visible
            StartXMLSpy_Click(null, null);

            // Open one of the sample files installed with the product.
            XMLSpy.Documents.OpenFile(strExamplesFolder + "ExpReport.xml", false);
        }

        // Handler for the "Toggle View Mode" button
        private void toggleView_Click(object sender, EventArgs e)
        {
            // Make sure there's a running XMLSpy instance, and that it's visible
            StartXMLSpy_Click(null, null);

            // Iterate through all open documents and toggle view between Text View and
Authentic View
            foreach (XMLSpyLib.Document doc in XMLSpy.Documents)
                if (doc.CurrentViewMode == XMLSpyLib.SPYViewModes.spyViewText)
                    doc.SwitchViewMode(XMLSpyLib.SPYViewModes.spyViewAuthentic);
                else
                    doc.SwitchViewMode(XMLSpyLib.SPYViewModes.spyViewText);
        }

        // Handler for the "Shutdown XMLSpy" button
        // Shut down the application instance by explicitly releasing the COM object
        private void shutdownXMLSpy_Click(object sender, EventArgs e)
        {
            if (XMLSpy != null)
            {
                // Allow shutdown of XMLSpy by releasing the UI
                XMLSpy.Visible = false;

                // Explicitly release the COM object
                try
                {
                    while
(System.Runtime.InteropServices.Marshal.ReleaseComObject(XMLSpy) > 0) ;
                }
                finally
                {
                    // Disallow subsequent access to this object
                    XMLSpy = null;
                }
            }
        }

        // Handler for button "Validate"
        private void validate_Click(object sender, EventArgs e)
        {
            // COM errors are returned to C# as exceptions. We use a try/catch block to
handle them.
            try
            {
                // Method 'IsValid' is one of the few functions that uses output
parameters
                // Use 'object' type for these parameters
                object strErrorText = "";
                object nErrorNumber = 0;
```

```
                object errorData = null;

                if (!XMLSpy.ActiveDocument.IsValid(ref strErrorText, ref nErrorNumber,
ref errorData))
                {
                    // The COM call succeeded but the document is not valid
                    // A detailed description of the problem is returned in strErrorText,
nErrorNumber and errorData
                    listBoxMessages.Items.Add("Document " + XMLSpy.ActiveDocument.Name +
" is not valid.");
                    listBoxMessages.Items.Add("\tErrorText  : " + strErrorText);
                    listBoxMessages.Items.Add("\tErrorNumber: " + nErrorNumber);
                    listBoxMessages.Items.Add("\tElement    : " + (errorData != null ?
((XMLSpyLib.XMLData)errorData).TextValue : "null"));
                }
                else
                {
                    // The COM call succeeded and the document is valid
                    listBoxMessages.Items.Add("Document " + XMLSpy.ActiveDocument.Name +
" is valid.");
                }
            }
            catch (Exception ex)
            {
                // The COM call was not successful
                // Probably no application instance has been started or no document is
open.
                listBoxMessages.Items.Add("Error validating active document: " +
ex.Message);
            }
        }



        // Event handler for OnDocumentOpened event
        private void handleOnDocumentOpened(XMLSpyLib.Document i_ipDocument)
        {
            MessageBox.Show("Document " + i_ipDocument.Name + " was opened!");
        }

        // Remember if the event handler is currently registered.
        private bool bEventHandlerIsRegistered = false;

        // Handler for button 'OnDocuemntOpened Event On/Off
        private void toggleOnDocumentOpenedEvent_Click(object sender, EventArgs e)
        {
            if (XMLSpy != null)
            {
                if (bEventHandlerIsRegistered)
                    XMLSpy.OnDocumentOpened -= new
XMLSpyLib._IApplicationEvents_OnDocumentOpenedEventHandler(handleOnDocumentOpened);
                else
                    XMLSpy.OnDocumentOpened += new
XMLSpyLib._IApplicationEvents_OnDocumentOpenedEventHandler(handleOnDocumentOpened);

                bEventHandlerIsRegistered = !bEventHandlerIsRegistered;
            }
        }
    }
}
```

25.3.1.2.3.1 Add Reference to XMLSpy API

Add the application's type library as a reference in a .NET project as follows: With the .NET project open, click **Project | Add Reference**. Then browse for the type library, which is called `XMLSpy.tlb`, and is located in the XMLSpy application folder.

Then declare a variable to access the XMLSpy API:

```
// An instance of XMLSpy is accessed via its automation interface.
   XMLSpyLib.Application XMLSpy;
```

25.3.1.2.3.2 Application Startup and Shutdown

In the code snippets below, the methods `StartXMLSpy_Click` and `ShutdownXMLSpy_Click` are those assigned to buttons in the AutomateXMLSpy example 1109 that, respectively, start up and shut down the application. This example is located in the C# subfolder of the API Examples folder (*see the file Form1.cs*):

Windows 7、Windows 8、Windows 10	`C:\Users\<username>\Documents\Altova\XMLSpy\2021\Examples`

You can compile and run the project from within Visual Studio 2010/2012/2013/2015/2017/2019.

Starting XMLSpy

The following code snippet from the AutomateXMLSpy example 1109 shows how to start up the application.

```
// Handler for the "Start XMLSpy" button
private void StartXMLSpy_Click(object sender, EventArgs e)
{
    if (XMLSpy == null)
    {
        Cursor.Current = Cursors.WaitCursor;

        // If no XMLSpy instance is running, we create one and make it visible
        XMLSpy = new XMLSpyLib.Application();
        XMLSpy.Visible = true;

        Cursor.Current = Cursors.Default;
    }
    else
    {
        // If an instance of XMLSpy is already running, make sure it's visible
        if (!XMLSpy.Visible)
            XMLSpy.Visible = true;
    }
}
```

Shutting down XMLSpy

The following code snippet from the AutomateXMLSpy example 1109 shows how to shut down the application.

```
// Handler for the "Shutdown XMLSpy" button
```

```
        // Shut down the application instance by explicitly releasing the COM object
         private void shutdownXMLSpy_Click(object sender, EventArgs e)
         {
             if (XMLSpy != null)
             {
                 // Allow shutdown of XMLSpy by releasing the UI
                 XMLSpy.Visible = false;

                 // Explicitly release COM object
                 try
                 {
                     while
(System.Runtime.InteropServices.Marshal.ReleaseComObject(XMLSpy) > 0) ;
                 }
                 finally
                 {
                     // Disallow subsequent access to this object
                     XMLSpy = null;
                 }
             }
         }
```

25.3.1.2.3.3 Opening Documents

The code snippets below (from the AutomateXMLSpy example[1109]) show how two files are opened via two separate methods assigned to two buttons in the user interface. Both methods use the same Application API access mechanism: XMLSpy.Documents.OpenFile(string, boolean)[1249].

The AutomateXMLSpy example[1109] (*see the file Form1.cs*) is located in the C# subfolder of the API Examples folder:

Windows 7、Windows 8、Windows 10	`C:\Users\<username>\Documents\Altova\XMLSpy\2021\Examples`

You can compile and run the project from within Visual Studio 2010/2012/2013/2015/2017/2019.

Code snippet

```
        // Handler for the "Open OrgChart.pxf" button
         private void openOrgChart_Click(object sender, EventArgs e)
         {
             // Make sure there's a running XMLSpy instance, and that it's visible
             StartXMLSpy_Click(null, null);

             // Open a file from the Examples folder installed with the product
             XMLSpy.Documents.OpenFile(strExamplesFolder + "OrgChart.pxf", false);
         }

         // Handler for the "Open ExpReport.xml" button
         private void openExpReport_Click(object sender, EventArgs e)
         {
             // Make sure there's a running XMLSpy instance, and that it's visible
             StartXMLSpy_Click(null, null);
```

```
            // Open a file from the Examples folder installed with the product
            XMLSpy.Documents.OpenFile(strExamplesFolder + "ExpReport.xml", false);
        }
```

The file opened last will be the active file.

25.3.1.2.3.4 Iterating through Open Documents

The code snippet below (from the AutomateXMLSpy example[1109]; *see the file Form1.cs*) shows how to iterate through open documents. A condition is then tested within the iteration loop, and the document view is switched between Text View and Authentic View.

```
       // Handler for the "Toggle view mode" button
        private void toggleView_Click(object sender, EventArgs e)
        {
            // Make sure there's a running XMLSpy instance, and that it's visible
            StartXMLSpy_Click(null, null);

            // Iterate through open documents and toggle current view between text and
authentic view.
            foreach (XMLSpyLib.Document doc in XMLSpy.Documents)
                if (doc.CurrentViewMode == XMLSpyLib.SPYViewModes.spyViewText)
                    doc.SwitchViewMode(XMLSpyLib.SPYViewModes.spyViewAuthentic);
                else
                    doc.SwitchViewMode(XMLSpyLib.SPYViewModes.spyViewText);
        }
```

The AutomateXMLSpy example[1109] example is located in the C# subfolder of the API Examples folder:

Windows 7、Windows 8、Windows 10	`C:\Users\<username>\Documents\Altova\XMLSpy\2021\Examples`

You can compile and run the project from within Visual Studio 2010/2012/2013/2015/2017/2019.

25.3.1.2.3.5 Errors and COM Output Parameters

The code snippet below (from the AutomateXMLSpy example[1109]) shows how to handle errors and COM output parameters. The method XMLSpy.ActiveDocument.IsValid(ref strErrorText, ref nErrorNumber, ref errorData)[1236] uses output parameters that are used, in the code snippet below, to generate an error-message text.

The AutomateXMLSpy example[1109] (*see the file Form1.cs*) is located in the C# subfolder of the API Examples folder:

Windows 7、Windows 8、Windows 10	`C:\Users\<username>\Documents\Altova\XMLSpy\2021\Examples`

You can compile and run the project from within Visual Studio 2010/2012/2013/2015/2017/2019.

Code snippet

```
        // Handler for button "Validate"
        private void validate_Click(object sender, EventArgs e)
        {
            // COM errors are returned to C# as exceptions. We use a try/catch block to
handle them.
            try
            {
                // Method 'IsValid' is one of the few functions that uses output
parameters
                // Use 'object' type for these parameters
                object strErrorText = "";
                object nErrorNumber = 0;
                object errorData = null;

                if (!XMLSpy.ActiveDocument.IsValid(ref strErrorText, ref nErrorNumber,
ref errorData))
                {
                    // The COM call succeeded but the document is not valid
                    // A detailed description of the problem is returned in strErrorText,
nErrorNumber and errorData
                    listBoxMessages.Items.Add("Document " + XMLSpy.ActiveDocument.Name +
" is not valid.");
                    listBoxMessages.Items.Add("\tErrorText  : " + strErrorText);
                    listBoxMessages.Items.Add("\tErrorNumber: " + nErrorNumber);
                    listBoxMessages.Items.Add("\tElement    : " + (errorData != null ?
((XMLSpyLib.XMLData)errorData).TextValue : "null"));
                }
                else
                {
                    // The COM call succeeded and the document is valid
                    listBoxMessages.Items.Add("Document " + XMLSpy.ActiveDocument.Name +
" is valid.");
                }
            }
            catch (Exception ex)
            {
                // The COM call was not successful
                // Probably no application instance has been started or no document is
open.
                listBoxMessages.Items.Add("Error validating active document: " +
ex.Message);
            }
        }
```

25.3.1.2.3.6 Events

The code snippet below (from the AutomateXMLSpy example[1109]) lists the code for two event handlers. The AutomateXMLSpy example[1109] (*see the file Form1.cs*) is located in the C# subfolder of the API Examples folder:

Windows 7、Windows 8、	C:\Users\<username>\Documents\

Windows 10	`Altova\XMLSpy\2021\Examples`

You can compile and run the project from within Visual Studio 2010/2012/2013/2015/2017/2019.

Code snippet

```
        // Event handler for OnDocumentOpened event
        private void handleOnDocumentOpened(XMLSpyLib.Document i_ipDocument)
        {
            MessageBox.Show("Document " + i_ipDocument.Name + " was opened!");
        }

        // Remember if the event handler is currently registered.
        private bool bEventHandlerIsRegistered = false;

        // Handler for button 'OnDocuemntOpened Event On/Off
        private void toggleOnDocumentOpenedEvent_Click(object sender, EventArgs e)
        {
            if (XMLSpy != null)
            {
                if (bEventHandlerIsRegistered)
                    XMLSpy.OnDocumentOpened -= new
XMLSpyLib._IApplicationEvents_OnDocumentOpenedEventHandler(handleOnDocumentOpened);
                else
                    XMLSpy.OnDocumentOpened += new
XMLSpyLib._IApplicationEvents_OnDocumentOpenedEventHandler(handleOnDocumentOpened);

                bEventHandlerIsRegistered = !bEventHandlerIsRegistered;
            }
        }
```

25.3.1.2.4 Java

The Application API can be accessed from Java code. To allow accessing the XMLSpy automation server directly from Java code, the libraries listed below must reside in the `classpath`. They are installed in the folder: `JavaAPI` in the XMLSpy application folder.

- `AltovaAutomation.dll`: a JNI wrapper for Altova automation servers (`AltovaAutomation_x64.dll` in the case of 64-bit versions)
- `AltovaAutomation.jar`: Java classes to access Altova automation servers
- `XMLSpyAPI.jar`: Java classes that wrap the XMLSpy automation interface
- `XMLSpyAPI_JavaDoc.zip`: a `Javadoc` file containing help documentation for the Java API

Note: In order to use the Java API, the DLL and Jar files must be on the Java Classpath.

Example Java project

An example Java project is supplied with your product installation. You can test the Java project and modify and use it as you like. For more details of the example Java project, see the section, Example Java Project 1120.

Rules for mapping the Application API names to Java

The rules for mapping between the Application API and the Java wrapper are as follows:

- ***Classes and class names***
  For every interface of the XMLSpy automation interface a Java class exists with the name of the interface.

- ***Method names***
  Method names on the Java interface are the same as used on the COM interfaces but start with a small letter to conform to Java naming conventions. To access COM properties, Java methods that prefix the property name with `get` and `set` can be used. If a property does not support write-access, no setter method is available. Example: For the `Name` property of the `Document` interface, the Java methods `getName` and `setName` are available.

- ***Enumerations***
  For every enumeration defined in the automation interface, a Java enumeration is defined with the same name and values.

- ***Events and event handlers***
  For every interface in the automation interface that supports events, a Java interface with the same name plus '`Event`' is available. To simplify the overloading of single events, a Java class with default implementations for all events is provided. The name of this Java class is the name of the event interface plus '`DefaultHandler`'. For example:
  `Application`: Java class to access the application
  `ApplicationEvents`: Events interface for the Application
  `ApplicationEventsDefaultHandler`: Default handler for `ApplicationEvents`

Exceptions to mapping rules

There are some exceptions to the rules listed above. These are listed below:

Interface	Java name
`Document`, method `SetEncoding`	`setFileEncoding`
`AuthenticView`, method `Goto`	`gotoElement`
`AuthenticRange`, method `Goto`	`gotoElement`
`AuthenticRange`, method `Clone`	`cloneRange`

This section

This section explains how some basic XMLSpy functionality can be accessed from Java code. It is organized into the following sub-sections:

- Example Java Project 1120
- Application Startup and Shutdown 1123
- Simple Document Access 1124
- Iterations 1124
- Use of Out-Parameters 1125
- Event Handlers 1125

25.3.1.2.4.1 Example Java Project

The XMLSpy installation package contains an example Java project, located in the the `API\Java` subfolder of the `Examples` folder :

Windows 7、Windows 8、Windows 10	`C:\Users\<username>\Documents\Altova\XMLSpy\2021\Examples`

This folder contains Java examples for the XMLSpy API. You can test it directly from the command line using the batch file `BuildAndRun.bat`, or you can compile and run the example project from within Eclipse. See below for instructions on how to use these procedures.

File list

The Java examples folder contains all the files required to run the example project. These files are listed below. If you are using a 64-bit version of the application, some filenames contain `_x64` in the name. These filenames are indicated with `(_x64)`.

`AltovaAutomation(_x64).dll`	Java-COM bridge: DLL part
`AltovaAutomation.jar`	Java-COM bridge: Java library part
`XMLSpyAPI.jar`	Java classes of the XMLSpy API
`RunXMLSpy.java`	Java example source code
`BuildAndRun.bat`	Batch file to compile and run example code from the command line prompt. Expects folder where Java Virtual Machine resides as parameter.
`.classpath`	Eclipse project helper file
`.project`	Eclipse project file
`XMLSpyAPI_JavaDoc.zip`	`Javadoc` file containing help documentation for the Java API

What the example does

The example starts up XMLSpy and performs a few operations, including opening and closing documents. When done, XMLSpy stays open. You must close it manually.

- Start XML Spy 1123: Starts XMLSpy, which is registered as an automation server, or activates XMLSpy if it is already running.
- Open OrgChart.pxf 1124: Locates one of the example documents installed with XMLSpy and opens it.
- Iteration and Changing the View Mode 1124: Changes the view of all open documents to Text View. The code also shows how to iterate through open documents.
- Iteration, validation, output parameters 1125: Validates the active document and shows the result in a message box. The code shows how to use output parameters.
- Event Handling 1125: Shows how to handle XMLSpy events.
- Shut down XMLSpy 1123: Shuts down XMLSpy.

You can modify the example in any way you like and run it.

Running the example from the command line

To run the example from the command line, open a command prompt window, go to the Java folder of the API Examples folder (*see above for location*), and then type:

```
buildAndRun.bat "<Path-to-the-Java-bin-folder>"
```

The Java binary folder must be that of a JDK 1.5 or later installation on your computer. Press the **Return** key. The Java source in `RunXMLSpy.java` will be compiled and then executed.

Loading the example in Eclipse

Open Eclipse and use the **Import | Existing Projects into Workspace** command to add the Eclipse project file (`.project`) located in the Java folder of the API Examples folder (*see above for location*). The project `RunXMLSpy` will then appear in your Package Explorer or Navigator. Select the project and then the command **Run as | Java Application** to execute the example.

Note: You can select a class name or method of the Java API and press F1 to get help for that class or method.

Java source code listing

The Java source code in the example file `RunXMLSpy.java` is listed below with comments.

```
001 // Access general JAVA-COM bridge classes
002 import com.altova.automation.libs.*;
003
004 // Access XMLSpy Java-COM bridge
005 import com.altova.automation.XMLSpy.*;
006 import com.altova.automation.XMLSpy.Enums.SPYViewModes;
007
008 /**
009  * An example that starts XMLSpy COM server and performs view operations on it
010  * Feel free to extend
011  */
012 public class RunXMLSpy
013 {
014   public static void main(String[] args)
015   {
016     // An instance of the application.
017     Application xmlSpy = null;
018
019     // Instead of COM error handling, use Java exception mechanism
020     try
021       {
022       // Start XMLSpy as COM server
023       xmlSpy = new Application();
024
025       // COM servers start up invisible, so make it visible
026       xmlSpy.setVisible(true);
027
028       // Locate samples installed with the product
029       String strExamplesFolder =
030       System.getenv("USERPROFILE") + "\\My Documents\\Altova\\XMLSpy2012\\Examples\
\";
031
032       // Open two example files
033       xmlSpy.getDocuments().openFile(strExamplesFolder + "OrgChart.pxf", false);
034       xmlSpy.getDocuments().openFile(strExamplesFolder + "ExpReport.xml", false);
035
036       // Iterate through open documents and set view mode to 'Text'.
037       for (Document doc:xmlSpy.getDocuments())
```

```
038       if ( doc.getCurrentViewMode() != SPYViewModes.spyViewText)
039           doc.switchViewMode(SPYViewModes.spyViewText);
040
041       // An alternative iteration mode is index-based
042       // COM indices are typically zero-based
043       Documents documents = xmlSpy.getDocuments();
044       for (int i = 1; i <= documents.getCount();
045           i++)
046           {
047             Document doc = documents.getItem(i);
048
049       // Validation is one of the few methods to have output parameters.
050       // The class JVariant is the correct type for parameters in these cases.
051       // To get values back mark them with the by-reference flag.
052       JVariant validationErrorText = new
053
054       JVariant.JStringVariant("");
055
056       validationErrorText.setByRefFlag();
057       JVariant validationErrorCount = new
058
059       JVariant.JIntVariant(0);
060
061       validationErrorCount.setByRefFlag();
062       JVariant validationErrorXMLData = new
063
064       JVariant.JIDispatchVariant(0);
065
066       validationErrorXMLData.setByRefFlag();
067       if (!doc.isValid(validationErrorText, validationErrorCount,
validationErrorXMLData))
068           System.out.println("Document" + doc.getName() + " is not wellformed - " +
validationErrorText.getStringValue());
069         else
070           System.out.println("Document" + doc.getName() + " is wellformed.");
071       }
072
073       // The following lines attach to the document events using a default
implementation
074       // for the events and override one of its methods.
075       // If you want to override all document events it is better to derive your
listener class
076       // from DocumentEvents and implement all methods of this interface.
077       Document doc = xmlSpy.getActiveDocument();
078       doc.addListener(new
079
080 DocumentEventsDefaultHandler()
081       {
082         @Override
083         public boolean
084
085 onBeforeCloseDocument(Document i_ipDoc) throws AutomationException
086         {
087           System.out.println("Document
088
089 " + i_ipDoc.getName() + " requested closing.");
090
091           // Allow closing of document
092           return true;
093         }
094       });
095       doc.close(true);
```

```
096       doc = null;
097
098       System.out.println("Watch XMLSpy!");
099     }
100     catch (AutomationException e)
101     {
102       // e.printStackTrace();
103     }
104     finally
105     {
106       // Make sure that XMLSpy can shut down properly.
107       if (xmlSpy != null)
108           xmlSpy.dispose();
109
110       // Since the COM server was made visible and still is visible,
111       // it will keep running, and needs to be closed manually.
112       System.out.println("Now close XMLSpy!");
113     }
114   }
115 }
```

25.3.1.2.4.2 Application Startup and Shutdown

The code listings below show how the application can be started up and shut down.

Application startup

Before starting up the application, the appropriate classes must be imported (*see below*).

```
01 // Access general JAVA-COM bridge classes
02 import com.altova.automation.libs.*;
03
04 // Access XMLSpy Java-COM bridge
05 import com.altova.automation.XMLSpy.*;
06 import com.altova.automation.XMLSpy.Enums.SPYViewModes;
07
08 /**
09  * An example that starts XMLSpy COM server and performs view operations on it
10  * Feel free to extend
11  */
12 public class RunXMLSpy
13 {
14   public static void main(String[] args)
15   {
16     // An instance of the application.
17     Application xmlSpy = null;
18
19     // Instead of COM error handling, use Java exception mechanism
20     try
21     {
22         // Start XMLSpy as COM server
23           xmlSpy = new Application();
24         // COM servers start up invisible, so make it visible
25           xmlSpy.setVisible(true);
26
27 ...
28     }
29   }
```

```
30 }
```

Application shutdown

The application can be shut down as shown below.

```
01 {
02       // Allow shutdown of XMLSpy by releasing the UI.
03       xmlSpy.setVisible(true);
04
05       // Make sure that XMLSpy can shut down properly.
06       if (xmlSpy != null)
07         xmlSpy.dispose();
08
09       // Since the COM server was made visible and still is visible,
10       // it will keep running, and needs to be closed manually.
11       System.out.println("Now close XMLSpy!");
12 }
```

25.3.1.2.4.3 Simple Document Access

The code listing below shows how to open a document.

```
1  // Locate samples installed with the product
2  String strExamplesFolder =
3  System.getenv("USERPROFILE") + "\\My Documents\\Altova\\XMLSpy2012\\Examples\\";
4
5
6  // Open file
7  xmlSpy.getDocuments().openFile(strExamplesFolder + "OrgChart.pxf", false);
```

25.3.1.2.4.4 Iterations

The listing below shows how to iterate through open documents.

```
01  // Iterate through open documents and set view mode to 'Text'.
02  for (Document doc:xmlSpy.getDocuments())
03    if ( doc.getCurrentViewMode() != SPYViewModes.spyViewText)
04            doc.switchViewMode(SPYViewModes.spyViewText);
05
06  // An alternative iteration mode is index-based
07  // COM indices are typically zero-based
08  Documents documents = xmlSpy.getDocuments();
09     for (int i = 1; i <= documents.getCount();
10         i++)
11       {
12         Document doc = documents.getItem(i);
13               ...
14       }
```

25.3.1.2.4.5 Use of Out-Parameters

The code listing below iterates through open documents and validates each of them. For each validation, a message is generated using the output parameters of the Validation method.

```
01 // An alternative iteration mode is index-based
02 // COM indices are typically zero-based
03 Documents documents = xmlSpy.getDocuments();
04 for (int i = 1; i <= documents.getCount();
05       i++)
06     {
07         Document doc = documents.getItem(i);
08
09 // Validation is one of the few methods to have output parameters.
10 // The class JVariant is the correct type for parameters in these cases.
11 // To get values back mark them with the by-reference flag.
12 JVariant validationErrorText = new
13
14 JVariant.JStringVariant("");
15
16 validationErrorText.setByRefFlag();
17       JVariant validationErrorCount = new
18
19 JVariant.JIntVariant(0);
20
21 validationErrorCount.setByRefFlag();
22       JVariant validationErrorXMLData = new
23
24 JVariant.JIDispatchVariant(0);
25
26 validationErrorXMLData.setByRefFlag();
27       if (!doc.isValid(validationErrorText,
28
29         validationErrorCount, validationErrorXMLData))
30         System.out.println("Document
31
32         " + doc.getName() + " is not wellformed - " +
33
34         validationErrorText.getStringValue());
35       else
36         System.out.println("Document
37
38         " + doc.getName() + " is wellformed.");
39     }
```

25.3.1.2.4.6 Event Handlers

The listing below shows how to listen for and use events.

```
01 // The following lines attach to the document events using a default implementation
02 // for the events and override one of its methods.
03 // If you want to override all document events it is better to derive your listener
class
04 // from DocumentEvents and implement all methods of this interface.
```

```
05
06 Document doc = xmlSpy.getActiveDocument();
07 doc.addListener(new DocumentEventsDefaultHandler()
08     {
09       @Override
10       public boolean
11       onBeforeCloseDocument(Document i_ipDoc) throws AutomationException
12       {
13         System.out.println("Document " + i_ipDoc.getName() + " requested closing.");
14
15         // Allow closing of document
16         return true;
17       }
18     });
19 doc.close(true);
20 doc = null;
```

25.3.1.3 The DOM and XMLData

The `XMLData` interface gives you full access to the XML structure behind the current document with less methods than DOM and is much simpler. The `XMLData` interface is a minimalist approach to reading and modifying existing, or newly created XML data. You might however, want to use a DOM tree because you can access one from an external source or you just prefer the MSXML DOM implementation.

The `ProcessDOMNode()` and `ProcessXMLDataNode()` functions provided below convert any segments of an XML structure between XMLData and DOM.

To use the `ProcessDOMNode()` function:

- pass the root element of the DOM segment you want to convert in objNode and
- pass the plugin object with the CreateChild() method in objCreator

To use the `ProcessXMLDataNode()` function:

- pass the root element of the XMLData segment in objXMLData and
- pass the DOMDocument object created with MSXML in xmlDoc

```
//////////////////////////////////////////////////////////////
// DOM To XMLData conversion
Function ProcessDOMNode(objNode,objCreator)
{
       var objRoot;
       objRoot = CreateXMLDataFromDOMNode(objNode,objCreator);

       If(objRoot)   {
              If((objNode.nodeValue != Null) && (objNode.nodeValue.length > 0))
                     objRoot.TextValue = objNode.nodeValue;
                     // add attributes
              If(objNode.attributes) {
                     var Attribute;
                     var oNodeList = objNode.attributes;

                     For(var i = 0;i < oNodeList.length; i++) {
                            Attribute = oNodeList.item(i);

                            var newNode;
                            newNode = ProcessDOMNode(Attribute,objCreator);
```

```
                        objRoot.AppendChild(newNode);
                }
        }
        If(objNode.hasChildNodes)  {
                try {
                        // add children
                        var Item;
                        oNodeList = objNode.childNodes;

                        For(var i = 0;i < oNodeList.length; i++) {
                                Item = oNodeList.item(i);

                                var newNode;
                                newNode = ProcessDOMNode(Item,objCreator);

                                objRoot.AppendChild(newNode);
                        }
                }
                catch(err) {
                }
        }
    }
    Return objRoot;
}

Function CreateXMLDataFromDOMNode(objNode,objCreator)
{
    var bSetName = True;
    var bSetValue = True;

    var nKind = 4;

    switch(objNode.nodeType)   {
        Case 2:nKind = 5;break;
        Case 3:nKind = 6;bSetName = False;break;
        Case 4:nKind = 7;bSetName = False;break;
        Case 8:nKind = 8;bSetName = False;break;
        Case 7:nKind = 9;break;
    }
    var objNew = Null;
    objNew = objCreator.CreateChild(nKind);

    If(bSetName)
        objNew.Name = objNode.nodeName;

    If(bSetValue && (objNode.nodeValue != Null))
        objNew.TextValue = objNode.nodeValue;

    Return objNew;
}
//////////////////////////////////////////////////////////////////
// XMLData To DOM conversion

Function ProcessXMLDataNode(objXMLData,xmlDoc)
{
    var objRoot;
    objRoot = CreateDOMNodeFromXMLData(objXMLData,xmlDoc);

    If(objRoot)   {
        If(IsTextNodeEnabled(objRoot) && (objXMLData.TextValue.length > 0))
            objRoot.appendChild(xmlDoc.createTextNode(objXMLData.TextValue));
```

```
            If(objXMLData.HasChildren) {
                  try {
                        var objChild;
                        objChild = objXMLData.GetFirstChild(-1);

                        While(True)   {
                              If(objChild)  {
                                    var newNode;
                                    newNode = ProcessXMLDataNode(objChild,xmlDoc);

                                    If(newNode.nodeType == 2)  {
                                          // child node is an attribute
                                          objRoot.attributes.setNamedItem(newNode);
                                    }
                                    Else
                                          objRoot.appendChild(newNode);
                              }
                              objChild = objXMLData.GetNextChild();
                        }
                  }
                  catch(err) {
                  }
            }
      }
      Return objRoot;
}

Function CreateDOMNodeFromXMLData(objXMLData,xmlDoc)
{
      switch(objXMLData.Kind)    {
            Case 4:Return xmlDoc.createElement(objXMLData.Name);
            Case 5:Return xmlDoc.createAttribute(objXMLData.Name);
            Case 6:Return xmlDoc.createTextNode(objXMLData.TextValue);
            Case 7:Return xmlDoc.createCDATASection(objXMLData.TextValue);
            Case 8:Return xmlDoc.createComment(objXMLData.TextValue);
            Case 9:Return
xmlDoc.createProcessingInstruction(objXMLData.Name,objXMLData.TextValue);
      }

      Return xmlDoc.createElement(objXMLData.Name);
}
Function IsTextNodeEnabled(objNode)
{
      switch(objNode.nodeType) {
            Case 1:
            Case 2:
            Case 5:
            Case 6:
            Case 11:Return True;
      }
      Return False;
}
```

25.3.1.4 Obsolete: Authentic View Row operations

If the schema on which an XML document is based specifies that an element is repeatable, such a structure can be represented in Authentic View as a table. When represented as a table, rows and their contents can be

manipulated individually, thereby allowing you to manipulate each of the repeatable elements individually. Such row operations would be performed by an external script.

If an external script is to perform row operations then two steps must occur:

- The first step checks whether the cursor is currently in a row using a property. Such a check could be, for example, `IsRowInsertEnabled`, which returns a value of either `TRUE` or `FALSE`.
- If the return value is `TRUE` then a row method, such as `RowAppend`, can be called. (`RowAppend` has no parameters and returns no value.)

The following is a list of properties and methods available for table operations. Each property returns a `BOOL`, and the methods have no parameter.

Property	Method	Table operations
IsRowInsertEnabled	**RowInsert** 1354, superseded by `AuthenticRange.InsertRow` 1172	Insert row operation
IsRowAppendEnabled	**RowAppend** 1353, superseded by `AuthenticRange.AppendRow` 1158	Append row operation
IsRowDeleteEnabled	**RowDelete** 1353, superseded by `AuthenticRange.DeleteRow` 1162	Delete row operation
IsRowMoveUpEnabled	**RowMoveUp** 1355, superseded by `AuthenticRange.MoveRowUp` 1179	Move XML data up one row
IsRowMoveDownEnabled	**RowMoveDown** 1355, superseded by `AuthenticRange.MoveRowDown` 1179	Move XML data down one row
IsRowDuplicateEnabled	**RowDuplicate** 1354, superseded by `AuthenticRange.DuplicateRow` 1162	Duplicate currently selected row

25.3.1.5 Obsolete: Authentic View Editing operations

When XML data is displayed as data in Authentic View, it is possible to manipulate individual elements using standard editing operations such as cut, copy, and paste. However, not all XML data nodes can be edited. So, in order to carry out an editing operation, first a property is used to test whether editing is possible, and then a method is called to perform the editing operation.

The only method that does not have a test is the method `EditSelectAll`, which automatically selects all elements displayed in the document.

The following is a list of properties and methods that perform editing operations. Each property returns a `BOOL`, and the methods have no parameter.

Property	Method	Editing operation
IsEditUndoEnabled	**EditUndo** 1341, superseded by `AuthenticView.Undo` 1202	Undo an editing operation
IsEditRedoEnabled	**EditRedo** 1340, superseded by `AuthenticView.Redo` 1201	Redo an editing operation

IsEditCopyEnabled	**EditCopy**[1339], superseded by AuthenticRange.Copy[1161]	Copy selected text to the clipboard
IsEditCutEnabled	**EditCut**[1339], superseded by AuthenticRange.Cut[1161]	Cut selected text to the clipboard
IsEditPasteEnabled	**EditPaste**[1340], superseded by AuthenticRange.Paste[1179]	Paste from clipboard to current cursor position
IsEditClearEnabled	**EditClear**[1338], superseded by AuthenticRange.Delete[1161]	Clear selected text from XML document

25.3.2 Interfaces

Object Hierarchy

Application[1131]
- SpyProject[1296]
  - SpyProjectItems[1300]
    - SpyProjectItem[1297]
- Documents[1246]
  - Document[1218]
    - GridView[1279]
    - AuthenticView[1185]
      - AuthenticRange[1157]
      - AuthenticDataTransfer[1152] (previously DocEditDataTransfer)
    - OldAuthenticView[1335] (previously DocEditView, **now obsolete**, superseded by AuthenticView[1185] and AuthenticRange[1157])
      - AuthenticSelection[1333] (previously DocEditSelection, **now obsolete**, superseded by AuthenticRange[1157])
      - AuthenticEvent[1321] (previously DocEditEvent, **now obsolete**)
      - AuthenticDataTransfer[1152] (previously DocEditDataTransfer)
    - TextView[1304]
    - XMLData[1309]
- Dialogs[1136]
  - CodeGeneratorDlg[1203]
  - FileSelectionDlg[1260]
  - SchemaDocumentationDlg[1282]
  - GenerateSampleXMLDlg[1273]
  - DTDSchemaGeneratorDlg[1251]
  - FindInFilesDlg[1261]
- DatabaseConnection[1209]
- ExportSettings[1257]
- TextImportExportSettings[1301]
- ElementList[1255]
  - ElementListItem[1256]

Enumerations[1358]

Description

This chapter contains the reference of the XMLSpy 1.5 Type Library.

(C) 2015-2021 Altova GmbH

Most of the given examples are written in VisualBasic. These code snippets assume that there is a variable defined and set, called **objSpy of type Application**. There are also some code samples written in JavaScript.

25.3.2.1 Application

Methods

GetDatabaseImportElementList 1137
GetDatabaseSettings 1138
GetDatabaseTables 1138
ImportFromDatabase 1141
CreateXMLSchemaFromDBStructure 1135

GetTextImportElementList 1139
GetTextImportExportSettings 1140
ImportFromText 1143

ImportFromWord 1144

ImportFromSchema 1142

GetExportSettings 1139

NewProject 1145
OpenProject 1145

AddMacroMenuItem 1134
ClearMacroMenu 1135

ShowForm 1148

ShowApplication 1148

URLDelete 1149
URLMakeDirectory 1149

AddXSLT_XQParameter 1135
GetXSLT_XQParameterCount 1140
GetXSLT_XQParameterName 1141
GetXSLT_XQParameterXPath 1141
RemoveXSLT_XQParameter 1146

FindInFiles 1137

Quit 1146

Properties

Application 1135
Parent 1145

ActiveDocument 1134
Documents 1136

CurrentProject 1136

Dialogs 1136

WarningNumber 1150
WarningText 1150

Status 1149
MajorVersion 1144
MinorVersion 1144
Edition 1137
IsAPISupported 1144
ServicePackVersion 1147

Description
Application is the root for all other objects. It is the only object you can create by CreateObject (VisualBasic) or other similar COM related functions.

Example

```
Dim objSpy As Application
Set objSpy = CreateObject("XMLSpy.Application")
```

25.3.2.1.1 Events

25.3.2.1.1.1 OnBeforeOpenDocument

Event: OnBeforeOpenDocument(*objDialog* as FileSelectionDlg 1260)

Description
This event gets fired whenever a document gets opened via the OpenFile or OpenURL menu command. It is sent after a document file has been selected but before the document gets opened. The file selection dialog object is initialized with the name of the selected document file. You can modify this selection. To continue the opening of the document leave the FileSelectionDlg.DialogAction 1260 property of *io_objDialog* at its default value spyDialogOK 1361. To abort the opening of the document set this property to spyDialogCancel 1361.

Examples
Given below are examples of how this event can be scripted.

XMLSpy scripting environment - VBScript:

```
Function On_BeforeOpenDocument(objDialog)
End Function
```

XMLSpy scripting environment - JScript:

```
function On_BeforeOpenDocument(objDialog)
{
}
```

XMLSpy IDE Plugin:

```
IXMLSpyPlugIn.OnEvent (26, ...)// nEventId = 26
```

25.3.2.1.1.2 OnBeforeOpenProject

Event: OnBeforeOpenProject(*objDialog* as FileSelectionDlg[1260])

Description
This event gets fired after a project file has been selected but before the project gets opened. The file selection dialog object is initialized with the name of the selected project file. You can modify this selection. To continue the opening of the project leave the FileSelectionDlg.DialogAction[1260] property of *io_objDialog* at its default value spyDialogOK[1361]. To abort the opening of the project set this property to spyDialogCancel[1361].

Examples
Given below are examples of how this event can be scripted.

XMLSpy scripting environment - VBScript:
Function On_BeforeOpenProject(*objDialog*)
End Function

XMLSpy scripting environment - JScript:
function On_BeforeOpenProject(*objDialog*)
{
}

XMLSpy IDE Plugin:
IXMLSpyPlugIn.OnEvent (25, ...)// nEventId = 25

25.3.2.1.1.3 OnDocumentOpened

Event: OnDocumentOpened(*objDocument* as Document[1218])

Description
This event gets fired whenever a document opens in XMLSpy. This can happen due to opening a file with the OpenFile or OpenURL dialog, creating a new file or dropping a file onto XMLSpy. The new document gets passed as parameter. The operation cannot be canceled.

Examples
Given below are examples of how this event can be scripted.

XMLSpy scripting environment - VBScript:
Function On_OpenDocument(*objDocument*)
End Function

XMLSpy scripting environment - JScript:
function On_OpenDocument(*objDocument*)
{
}

XMLSpy IDE Plugin:
IXMLSpyPlugIn.OnEvent (7, ...) // nEventId = 7

25.3.2.1.1.4 OnProjectOpened

Event: OnProjectOpened(*objProject* as SpyProject[1296])

Description
This event gets fired whenever a project gets opened in XMLSpy. The new project gets passed as parameter.

Examples
Given below are examples of how this event can be scripted.

XMLSpy scripting environment - VBScript:
Function On_OpenProject(*objProject*)
End Function

XMLSpy scripting environment - JScript:
function On_OpenProject(*objProject*)
{
}

XMLSpy IDE Plugin:
IXMLSpyPlugIn.OnEvent (6, ...) // nEventId = 6

25.3.2.1.2 ActiveDocument

Property: ActiveDocument as Document[1218]

Description
Reference to the active document. If no document is open, ActiveDocument is null (nothing).

Errors

1111	The application object is no longer valid.
1100	Invalid address for the return parameter was specified.

25.3.2.1.3 AddMacroMenuItem

Method: AddMacroMenuItem(*strMacro* as String,*strDisplayText* as String)

Description
Adds a menu item to the **Tools** menu. This new menu item invokes the macro defined by strMacro. See also Example Scripting Project[1067].

Errors

1111	The application object is no longer valid.
1100	Invalid parameter or invalid address for the return parameter was specified.
1108	Number of macro items is limited to 16 items.

(C) 2015-2021 Altova GmbH

25.3.2.1.4 AddXSLT_XQParameter

Method: AddXSLT_XQParameter(name as String, XPath as String)

Description
Adds an XSLT or XQuery parameter. The parameter's name and value are the two arguments of the method.

Errors
1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.
1124 The XPath expression is not set.
1125 Not a QName.
1126 The specified XPath is not valid. Reason for invalidity appended.
1127 A parameter with the submitted name already exists.

25.3.2.1.5 Application

Property: Application as Application 1131 (read-only)

Description
Accesses the XMLSpy application object.

Errors
1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

25.3.2.1.6 ClearMacroMenu

Method: ClearMacroMenu()

Return Value
None

Description
Removes from the **Tools** menu those menu items that were added by calling `AddMacroMenuItem` 1134. See also Example Scripting Project 1067.

Errors
1111 The application object is no longer valid.

25.3.2.1.7 CreateXMLSchemaFromDBStructure

Method: CreateXMLSchemaFromDBStructure(pImportSettings as DatabaseConnection 1209, pTables as ElementList 1255)

Description
CreateXMLSchemaFromDBStructure creates from a database specified in pImportSettings for the defined tables in pTables new XML Schema document(s) describing the database tables structure.

The parameter pTables specifies which table structures the XML Schema document should contain. This parameter can be NULL, specifying that all table structures will be exported.

See also GetDataBaseTables 1138.

Errors
1112 Invalid database specified.
1120 Database import failed.

25.3.2.1.8 CurrentProject

Property: CurrentProject as SpyProject 1296

Description
Reference to the active document. If no project is open, CurrentProject is null (nothing).

Errors
1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

25.3.2.1.9 Dialogs

Property: Dialogs as Dialogs 1215 (read-only)

Description
Access the built-in dialogs of XMLSpy.

Errors
1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

25.3.2.1.10 Documents

Property: Documents as Documents 1246

Description
Collection of all open documents.

Errors
1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

(C) 2015-2021 Altova GmbH

25.3.2.1.11 Edition

Property: Edition as String

Description
Returns the edition of the application, for example `Altova XMLSpy Enterprise Edition` for the Enterprise edition.

Errors
1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

25.3.2.1.12 FindInFiles

Method: FindInFiles(*pSettings* as FindInFilesDlg[1261]) as FindInFilesResults[1272]

Description
Returns a FindInFilesResults[1272] object containing information about the files that matched the specified settings.

Errors
1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

25.3.2.1.13 GetDatabaseImportElementList

Method: GetDatabaseImportElementList(*pImportSettings* as DatabaseConnection[1209]) as ElementList[1255]

Description
The function returns a collection of ElementListItems where the properties ElementListItem.Name[1257] contain the names of the fields that can be selected for import and the properties ElementListItem.ElementKind[1256] are initialized either to *spyXMLDataAttr* or *spyXMLDataElement*, depending on the value passed in DatabaseConnection.AsAttributes[1210]. This list serves as a filter to what finally gets imported by a future call to ImportFromDatabase[1141]. Use ElementList.RemoveElement[1256] to exclude fields from import.

Properties mandatory to be filled out for the database connection are one of DatabaseConnection.File[1212], DatabaseConnection.ADOConnection[1210] and DatabaseConnection.ODBCConnection[1213], as well as DatabaseConnection.SQLSelect[1214]. Use the property DatabaseConnection.AsAttributes[1210] to initialize ElementListItem.ElementKind[1256] of the resulting element list to either *spyXMLDataAttr* or *spyXMLDataElement*, respectively.

Example
See example at ImportFromDatabase[1141].

Errors

1111 The application object is no longer valid.
1100 Invalid parameter or invalid address for the return parameter was specified.
1107 Import from database failed.
1112 Invalid database specified.
1114 Select statement is missing.
1119 database element list import failed.

25.3.2.1.14 GetDatabaseSettings

Method: GetDatabaseSettings() as DatabaseConnection[1209]

Description
GetDatabaseSettings creates a new object of database settings. The object is used to specify database connection parameters for the methods GetDatabaseTables[1138], GetDatabaseImportElementList[1137], ImportFromDatabase[1141], ImportFromSchema[1142] and ExportToDatabase[1230].

Example
See example of ImportFromDatabase[1141].

Errors
1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

25.3.2.1.15 GetDatabaseTables

Method: GetDatabaseTables(*pImportSettings* as DatabaseConnection[1131]) as ElementList[1255]

Description
GetDatabaseTables reads the table names from the database specified in *pImportSettings*. Properties mandatory to be filled out for the database connection are one of DatabaseConnection.File[1212], DatabaseConnection.ADOConnection[1210] and DatabaseConnection.ODBCConnection[1213]. All other properties are ignored.
The function returns a collection of ElementListItems where the properties ElementListItem.Name[1257] contain the names of tables stored in the specified database. The remaining properties of ElementListItem[1256] are unused.

Errors
1111 The application object is no longer valid.
1100 Invalid parameter or invalid address for the return parameter was specified.
1112 Invalid database specified.
1113 Error while reading database table information.
1118 Database table query failed.

Example

```
Dim objImpSettings As DatabaseConnection
Set objImpSettings = objSpy.GetDatabaseSettings
```

```
objImpSettings.ADOConnection = TxtADO.Text

'store table names in list box
ListTables.Clear

Dim objList As ElementList
Dim objItem As ElementListItem
On Error GoTo ErrorHandler
Set objList = objSpy.GetDatabaseTables(objImpSettings)

        For Each objItem In objList
                ListTables.AddItem objItem.Name
        Next
```

25.3.2.1.16 GetExportSettings

Method: GetExportSettings()as ExportSettings[1257] (read-only)

Description
GetExportSettings creates a new object of common export settings. This object is used to pass the parameters to the export functions and defines the behaviour of the export calls. See also the export functions from Document[1218] and the examples at Import and Export[1103].

Errors

1111	The application object is no longer valid.
1100	Invalid address for the return parameter was specified.

25.3.2.1.17 GetTextImportElementList

Method: GetTextImportElementList(pImportSettings as TextImportExportSettings[1301]) as ElementList[1255]

Description
GetTextImportElementList retrieves importing information about the text-file as specified in pImportSettings. The function returns a collection of ElementListItems where the properties ElementListItem.Name[1257] contain the names of the fields found in the file. The values of remaining properties are undefined.

If the text-file does not contain a column header, set `pImportSettings.HeaderRow`[1303] to `false`. The resulting element list will contain general column names like 'Field1' and so on.

See also Import and export of data[1103].

Errors

1111	The application object is no longer valid.
1100	Invalid parameter or invalid address for the return parameter was specified.
1107	Import from database failed.
1115	Error during text element list import. Cannot create parser for import file.
1116	Error during text element list import.

Example

```
' -------------------------------------------------------
' VBA client code fragment - import selected fields from text file
' -------------------------------------------------------
        Dim objImpSettings As TextImportExportSettings
        Set objImpSettings = objSpy.GetTextImportExportSettings

        objImpSettings.ImportFile = "C:\ImportMe.txt"
        objImpSettings.HeaderRow = False

        Dim objList As ElementList
        Set objList = objSpy.GetTextImportElementList(objImpSettings)

        'exclude first column
        objList.RemoveItem 1

        Dim objImpDoc As Document
        On Error Resume Next
        Set objImpDoc = objSpy.ImportFromText(objImpSettings, objList)
        CheckForError
```

25.3.2.1.18 GetTextImportExportSettings

Method: GetTextImportExportSettings() as TextImportExportSettings[1301] (read-only)

Description
GetTextImportExportSettings creates a new object of common import and export settings for text files. See also the example for Application.GetTextImportElementList[1139] and Import and Export[1103].

Errors
1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

25.3.2.1.19 GetXSLT_XQParameterCount

Method: GetXSLT_XQParameterCount() as Long

Description
Returns the number of XSLT and XQuery parameters.

Errors
1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

(C) 2015-2021 Altova GmbH

25.3.2.1.20 GetXSLT_XQParameterName

Method: GetXSLT_XQParameterName(index as Long) as String

Description
Returns the name of the XSLT or XQuery parameter identified by the supplied index.

Errors
1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

25.3.2.1.21 GetXSLT_XQParameterXPath

Method: GetXSLT_XQParameterXPath(index as Long) as String

Description
Returns the XPath expression of the XSLT or XQuery parameter identified by the supplied index.

Errors
1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

25.3.2.1.22 ImportFromDatabase

Method: ImportFromDatabase(pImportSettings as DatabaseConnection[1209],pElementList as ElementList[1255]) as Document[1218]

Return Value
Creates a new document containing the data imported from the database.

Description
ImportFromDatabase imports data from a database as specified in pImportSettings and creates a new document containing the data imported from the database. Properties mandatory to be filled out are one of DatabaseConnection.File[1212], DatabaseConnection.ADOConnection[1210] or DatabaseConnection.ODBCConnection[1213] and DatabaseConnection.SQLSelect[1214]. Additionally, you can use DatabaseConnection.AsAttributes[1210], DatabaseConnection.ExcludeKeys[1212], DatabaseConnection.IncludeEmptyElements[1213] and NumberDateTimeFormat[1213] to further parameterize import.

The parameter pElementList specifies which fields of the selected data gets written into the newly created document, and which are created as elements and which as attributes. This parameter can be NULL, specifying that all selected fields will be imported as XML elements.

See GetDatabaseSettings[1138] and GetDatabaseImportElementList[1137] for necessary steps preceding any import of data from a database.

Errors

1111 The application object is no longer valid.
1100 Invalid parameter or invalid address for the return parameter was specified.
1107 Import from database failed.
1112 Invalid database specified.
1114 Select statement is missing.
1117 Transformation to XML failed.
1120 Database import failed.

Example

```
Dim objImpSettings As DatabaseConnection
Set objImpSettings = objSpy.GetDatabaseSettings

objImpSettings.ADOConnection = strADOConnection
objImpSettings.SQLSelect = "SELECT * FROM MyTable"

Dim objDoc As Document
On Error Resume Next
Set objDoc = objSpy.ImportFromDatabase(objImpSettings,
objSpy.GetDatabaseImportElementList(objImpSettings))
' CheckForError here
```

25.3.2.1.23 ImportFromSchema

Method: ImportFromSchema(*pImportSettings* as DatabaseConnection[1209],*strTable* as String,*pSchemaDoc* as Document[1218]) as Document[1218]

Return Value
Creates a new document filled with data from the specified database as specified by the schema definition in *pSchemaDoc*.

Description
ImportFromSchema imports data from a database specified in pImportSettings. Properties mandatory to be filled out are one of DatabaseConnection.File[1212], DatabaseConnection.ADOConnection[1210] or DatabaseConnection.ODBCConnection[1213]. Additionally, you can use DatabaseConnection.AsAttributes[1210], DatabaseConnection.ExcludeKeys[1212] and NumberDateTimeFormat[1213] to further parameterize import. All other properties get ignored.

ImportFromSchema does not use an explicit SQL statement to select the data. Instead, it expects a structure definition of the document to create in form of an XML schema document in *pSchemaDoc*. From this definition the database select statement is automatically deduced. Specify in *strTable* the table name of the import root that will become the root node in the new document.

See GetDatabaseSettings[1138] and GetDatabaseTables[1138] for necessary steps preceding an import from a database based on a schema definition. To create the schema definition file use command 'create database schema' from the 'convert' menu of XMLSpy.

Errors
1111 The application object is no longer valid.

(C) 2015-2021 Altova GmbH

1100 Invalid parameter or invalid address for the return parameter was specified.
1107 Import from database failed.
1112 Invalid database specified.
1120 Database import failed.
1121 Could not create validator for the specified schema.
1122 Failed parsing schema for database import.

25.3.2.1.24 ImportFromText

Method: ImportFromText(*pImportSettings* as TextImportExportSettings[1301],*pElementList* as ElementList[1255]) as Document[1218]

Description
ImportFromText imports the text file as specified in pImportSettings. The parameter pElementList can be used as import filter. Either pass the list returned by a previous call to GetTextImportElementList[1139] or null to import all columns. To avoid import of unnecessary columns use ElementList.RemoveElement[1256] to remove the corresponding field names from `pElementList` before calling `ImportFromText`.
The method returns the newly created document containing the imported data. This document is the same as the active document of XMLSpy.

See also Import and export of data[1103].

Errors
1111 The application object is no longer valid.
1100 Invalid parameter or invalid address for the return parameter was specified.
1107 Import from text file failed.
1117 Transformation to XML failed.

Example

```
' -------------------------------------------------------
' VBA client code fragment - import from text file
' -------------------------------------------------------
        Dim objImpSettings As TextImportExportSettings
        Set objImpSettings = objSpy.GetTextImportExportSettings

        objImpSettings.ImportFile = strFileName
        objImpSettings.HeaderRow = False

        Dim objImpDoc As Document
        On Error Resume Next
        Set objImpDoc = objSpy.ImportFromText(objImpSettings,
objSpy.GetTextImportElementList(objImpSettings))

        CheckForError
```

25.3.2.1.25 ImportFromWord

Method: ImportFromWord(*strFile* as String) as Document[1218]

Description
ImportFromWord imports the MS-Word Document strFile into a new XML document.

Errors
1111 The application object is no longer valid.
1100 Invalid parameter or invalid address for the return parameter was specified.
Import from document failed.

25.3.2.1.26 IsAPISupported

Property: IsAPISupported as Boolean

Description
Returns whether the API is supported in this version or not.

Errors
1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

25.3.2.1.27 MajorVersion

Property: MajorVersion as Integer

Description
Returns the application version's major number, for example `15` for 2013 versions, and `16` for 2014 versions..

Errors
1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

25.3.2.1.28 MinorVersion

Property: MinorVersion as Integer

Description
Returns the application version's minor number.

Errors
1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

(C) 2015-2021 Altova GmbH

25.3.2.1.29 NewProject

Method: NewProject(*strPath* as String,*bDiscardCurrent* as Boolean)

Description
NewProject creates a new project.

If there is already a project open that has been modified and bDiscardCurrent is false, then NewProject() fails.

Errors
1111 The application object is no longer valid.
1102 A project is already open but *bDiscardCurrent* is *true*.
1103 Creation of new project failed.

25.3.2.1.30 OpenProject

Method: OpenProject(*strPath* as String,*bDiscardCurrent* as Boolean,*bDialog* as Boolean)

Parameters
strPath
Path and file name of the project to open. Can be empty if bDialog is true.

bDiscardCurrent
Discard currently open project and possibly lose changes.

bDialog
Show dialogs for user input.

Return Value
None

Description
OpenProject opens an existing project. If there is already a project open that has been modified and bDiscardCurrent is false, then OpenProject() fails.

Errors
1111 The application object is no longer valid.
1100 Invalid parameter or invalid address for the return parameter was specified.
1101 Cannot open specified project.
1102 A project is already open but *bDiscardCurrent* is *true*.

25.3.2.1.31 Parent

Property: Parent as Application 1131 (read-only)

Description

Accesses the XMLSpy application object.

Errors
1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

25.3.2.1.32 Quit

Method: Quit()

Return Value
None

Description
This method terminates XMLSpy. All modified documents will be closed without saving the changes. This is also true for an open project.

If XMLSpy was automatically started as an automation server by a client program, the application will not shut down automatically when your client program shuts down if a project or any document is still open. Use the Quit method to ensure automatic shut-down.

Errors
1111 The application object is no longer valid.

25.3.2.1.33 ReloadSettings

Method: ReloadSettings

Return Value

Description
The application settings are reloaded from the registry.

Available with TypeLibrary version 1.5

Errors
1111 The application object is no longer valid.

25.3.2.1.34 RemoveXSLT_XQParameter

Method: RemoveXSLT_XQParameter(index as Long)

Description
Removes the XSLT or XQuery parameter identified by the supplied index.

Errors

1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

25.3.2.1.35 RunMacro

Method: RunMacro(*strMacro* as String)

Return Value

Description
Calls the specified macro either from the project scripts (if present) or from the global scripts.

Available with TypeLibrary version 1.5

Errors
1111 The application object is no longer valid.

25.3.2.1.36 ScriptingEnvironment

Property: ScriptingEnvironment as IUnknown (read-only)

Description
Reference to any active scripting environment. This property makes it possible to access the TypeLibrary of the XMLSpyFormEditor.exe application which is used as the current scripting environment.

Available with TypeLibrary version 1.5

Errors
1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

25.3.2.1.37 ServicePackVersion

Property: ServicePackVersion as Long

Description
Returns the Service Pack version number of the application. Eg: 1 for 2010 R2 SP1

Errors
1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

25.3.2.1.38 ShowApplication

Method: ShowApplication(*bShow* as Boolean)

Return Value
None

Description
The method shows (bShow = True) or hides (bShow = False) XMLSpy.

Errors
1110 The application object is no longer valid.

25.3.2.1.39 ShowFindInFiles

Method: ShowFindInFiles(pSettings as FindInFilesDlg 1261) as Boolean

Return Value
Returns false if the user pressed the Cancel button, true otherwise.

Description
Displays the FindInFiles dialog preset with the given settings. The user modifications of the settings are stored in the passed dialog object.

Errors
1111 The application object is no longer valid.
1100 Invalid parameter or invalid address for the return parameter was specified.

25.3.2.1.40 ShowForm

Method: ShowForm(*strFormName* as String) as Long

Return Value
Returns zero if the user pressed a Cancel button or the form calls TheView.Cancel().

Description
Displays the form strFormName.

Forms, event handlers and macros can be created with the Scripting Environment. Select "Switch to scripting environment" from the **Tools** menu to invoke the Scripting Environment.

Errors
1111 The application object is no longer valid.
1100 Invalid parameter or invalid address for the return parameter was specified.

(C) 2015-2021 Altova GmbH

25.3.2.1.41 Status

Property: Status as ENUMApplicationStatus 1358

Description
Returns the current status of the running application.

Errors
1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

25.3.2.1.42 URLDelete

Method: URLDelete(*strURL* as String,*strUser* as String,*strPassword* as String)

Return Value
None

Description
The method deletes the file at the URL strURL.

Errors
1111 The application object is no longer valid.
1109 Error deleting file at specified URL.

25.3.2.1.43 URLMakeDirectory

Method: URLMakeDirectory(*strURL* as String,*strUser* as String,*strPassword* as String)

Return Value
None

Description
The method creates a new directory at the URL strURL.

Errors
1111 The application object is no longer valid.
1100 Invalid parameter specified.

25.3.2.1.44 Visible

Property: Visible as VARIANT_BOOL

Description
Sets or gets the visibility attribute of XMLSpy. This standard automation property makes usage of ShowApplication 1148 obsolete.

Errors
1110 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

25.3.2.1.45 WarningNumber

Property: WarningNumber as integer

Description
Some methods fill the property WarningNumber with additional information if an error occurs.

Currently just Documents.OpenFile 1249 fills this property.

Errors
1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

25.3.2.1.46 WarningText

Property: WarningText as String

Description
Some methods fill the property WarningText with additional information if an error occurs.

Currently just Documents.OpenFile 1249 fills this property.

Errors
1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

25.3.2.2 AuthenticContextMenu

The context menu interface provides the means for the user to customize the context menus shown in Authentic. The interface has the methods listed in this section.

25.3.2.2.1 CountItems

Method: CountItems() nItems as long

Return Value
Returns the number of menu items.

Errors
2501 Invalid object.

25.3.2.2.2 DeleteItem

Method: DeleteItem`(IndexPosition as long)`

Return Value
Deletes the menu item that has the index position submitted in the first parameter.

Errors
2501 Invalid object
2502 Invalid index

25.3.2.2.3 GetItemText

Method: GetItemText`(IndexPosition as long)` MenuItemName as string

Return Value
Gets the name of the menu item located at the index position submitted in the first parameter.

Errors
2501 Invalid object
2502 Invalid index

25.3.2.2.4 InsertItem

Method: InsertItem`(IndexPosition as long, MenuItemName as string, MacroName as string)`

Return Value
Inserts a user-defined menu item at the position in the menu specified in the first parameter and having the name submitted in the second parameter. The menu item will start a macro, so a valid macro name must be submitted.

Errors
2501 Invalid object
2502 Invalid index
2503 No such macro
2504 Internal error

25.3.2.2.5 SetItemText

Method: SetItemText`(IndexPosition as long, MenuItemName as string)`

Return Value
Sets the name of the menu item located at the index position submitted in the first parameter.

Errors
2501 Invalid object
2502 Invalid index

25.3.2.3 AuthenticDataTransfer

Renamed from DocEditDataTransfer to AuthenticDataTransfer

The DocEditView object is renamed to OldAuthenticView.
DocEditSelection is renamed to AuthenticSelection.
DocEditEvent is renamed to AuthenticEvent.
DocEditDataTransfer is renamed to AuthenticDataTransfer.

Their usage—except for AuthenticDataTransfer—is no longer recommended. We will continue to support existing functionality for a yet undefined period of time but no new features will be added to these interfaces. All functionality available up to now in DocEditView 1335, DocEditSelection 1333, DocEditEvent 1321 and DocEditDataTransfer 1153 is now available via AuthenticView 1185, AuthenticRange 1157 and AuthenticDataTransfer 1152. Many new features have been added.

For examples on migrating from DocEdit to Authentic see the description of the different methods and properties of the different DocEdit objects.

Methods

getData 1153

Properties

dropEffect 1153
ownDrag 1153
type 1153

Description

The events OnDragOver and OnBeforeDrop provide information about the object being dragged with an instance of type AuthenticDataTransfer. It contains a description of the dragged object and its content. The latter is available either as string or a pointer to a COM object supporting the IUnkown interface.

(C) 2015-2021 Altova GmbH

25.3.2.3.1 dropEffect

Property: dropEffect as long

Description
The property stores the drop effect from the default event handler. You can set the drop effect if you change this value and return TRUE for the event handler (or set AuthenticEvent.cancelBubble 1324 to TRUE if you are still using the now obsolete AuthenticEvent interface).

Errors
2101 Invalid address for the return parameter was specified.

25.3.2.3.2 getData

Method: getData() as Variant

Description
Retrieve the data associated with the dragged object. Depending on AuthenticDataTransfer.type 1153, that data is either a string or a COM interface pointer of type IUnknown.

Errors
2101 Invalid address for the return parameter was specified.

25.3.2.3.3 ownDrag

Property: ownDrag as Boolean (read-only)

Description
The property is TRUE if the current dragging source comes from inside Authentic View.

Errors
2101 Invalid address for the return parameter was specified.

25.3.2.3.4 type

Property: type as String (read-only)

Description
Holds the type of data you get with the DocEditDataTransfer.getData 1153 method.

Currently supported data types are:

OWN	data from Authentic View itself
TEXT	plain text
UNICODETEXT	plain text as UNICODE

Errors
2101 Invalid address for the return parameter was specified.

25.3.2.4 AuthenticEventContext

The `EventContext` interface gives access to many properties of the context in which a macro is executed.

25.3.2.4.1 EvaluateXPath

Method: EvaluateXPath (strExpression as string) as strValue as string

Return Value
The method evaluates the XPath expression in the context of the node within which the event was triggered and returns a string.

Description
`EvaluateXPath()` executes an XPath expression with the given event context. The result is returned as a string, in the case of a sequence it is a space-separated string.

Errors
2201 Invalid object.
2202 No context.
2209 Invalid parameter.
2210 Internal error.
2211 XPath error.

25.3.2.4.2 GetEventContextType

Method: GetEventContextType `()` Type as `AuthenticEventContextType` enumeration

Return Value
Returns the context node type.

Description
`GetEventContextType` allows the user to determine whether the macro is in an XML node or in an XPath atomic item context. The enumeration `AuthenticEventContextType` is defined as follows:

authenticEventContextXML,
authenticEventContextAtomicItem,
authenticEventContextOther

If the context is a normal XML node, the `GetXMLNode()` function gives access to it (returns `NULL` if not).

Errors
2201 Invalid object.
2202 No context.

(C) 2015-2021 Altova GmbH

2209 Invalid parameter.

25.3.2.4.3 GetNormalizedTextValue

Method: GetNormalizedTextValue() strValue as string

Return Value
Returns the value of the current node as string

Errors
2201 Invalid object.
2202 No context.
2203 Invalid context
2209 Invalid parameter.

25.3.2.4.4 GetVariableValue

Method: GetVariableValue(strName as string) strValue as string

Return Value
Gets the value of the variable submitted as the parameter.

Description
`GetVariableValue` gets the variable's value in the scope of the context.

```
nZoom = parseInt( AuthenticView.EventContext.GetVariableValue( 'Zoom' ) );
if ( nZoom > 1 )
{
    AuthenticView.EventContext.SetVariableValue( 'Zoom', nZoom - 1 );
}
```

Errors
2201 Invalid object.
2202 No context.
2204 No such variable in scope
2205 Variable cannot be evaluated
2206 Variable returns sequence
2209 Invalid parameter

25.3.2.4.5 GetXMLNode

Method: GetXMLNode() Node as XMLData object

Return Value
Returns the context XML node or `NULL`

Errors

2201 Invalid object.
2202 No context.
2203 Invalid context
2209 Invalid parameter.

25.3.2.4.6 IsAvailable

Method: IsAvailable`()` as Boolean

Return Value

Returns true if `EventContext` is set, false otherwise.

Errors

2201 Invalid object.

25.3.2.4.7 SetVariableValue

Method: SetVariableValue`(strName as string, strValue as string)`

Return Value

Sets the value (second parameter) of the variable submitted in the first parameter.

Description

`SetVariableValue` sets the variable's value in the scope of the context.

```
nZoom = parseInt( AuthenticView.EventContext.GetVariableValue( 'Zoom' ) );
if ( nZoom > 1 )
{
    AuthenticView.EventContext.SetVariableValue( 'Zoom', nZoom - 1 );
}
```

Errors

2201 Invalid object.
2202 No context.
2204 No such variable in scope
2205 Variable cannot be evaluated
2206 Variable returns sequence
2207 Variable read-only
2208 No modification allowed

(C) 2015-2021 Altova GmbH

25.3.2.5 AuthenticRange

The first table lists the properties and methods of AuthenticRange that can be used to navigate through the document and select specific portions.

Properties	Methods	
Application 1159	Clone 1160	MoveBegin 1178
FirstTextPosition 1164	CollapsToBegin 1160	MoveEnd 1178
FirstXMLData 1164	CollapsToEnd 1161	NextCursorPosition 1170
FirstXMLDataOffset 1165	ExpandTo 1163	PreviousCursorPosition 1171
LastTextPosition 1175	Goto 1169	Select 1181
LastXMLData 1176	GotoNext 1169	SelectNext 1181
LastXMLDataOffset 1177	GotoPrevious 1170	SelectPrevious 1182
Parent 1179	IsEmpty 1173	SetFromRange 1184
	IsEqual 1174	

The following table lists the content modification methods, most of which can be found on the right/button mouse menu.

Properties	Edit operations	Dynamic table operations
Text 1184	Copy 1161	AppendRow 1158
	Cut 1161	DeleteRow 1162
	Delete 1161	DuplicateRow 1162
	IsCopyEnabled 1173	InsertRow 1172
	IsCutEnabled 1173	IsFirstRow 1174
	IsDeleteEnabled 1173	IsInDynamicTable 1174
	IsPasteEnabled 1175	IsLastRow 1174
	Paste 1179	MoveRowDown 1179
		MoveRowUp 1179

The following methods provide the functionality of the Authentic entry helper windows for range objects.

Operations of the entry helper windows		
Elements	**Attributes**	**Entities**
CanPerformActionWith 1160	GetElementAttributeValue 1167	GetEntityNames 1168

CanPerformAction 1159	GetElementAttributeNames 1166	InsertEntity 1171
PerformAction 1180	GetElementHierarchy 1167	
	HasElementAttribute 1171	
	IsTextStateApplied 1175	
	SetElementAttributeValue 1183	

Description
AuthenticRange objects are the 'cursor' selections of the automation interface. You can use them to point to any cursor position in the Authentic view, or select a portion of the document. The operations available for AuthenticRange objects then work on this selection in the same way, as the corresponding operations of the user interface do with the current user interface selection. The main difference is that you can use an arbitrary number of AuthenticRange objects at the same time, whereas there is exactly one cursor selection in the user interface.

To get to an initial range object use AuthenticView.Selection 1201, to obtain a range corresponding with the current cursor selection in the user interface. Alternatively, some trivial ranges are accessible via the read/only properties AuthenticView.DocumentBegin 1196, AuthenticView.DocumentEnd 1196, and AuthenticView.WholeDocument 1203. The most flexible method is AuthenticView.Goto 1198, which allows navigation to a specific portion of the document within one call. For more complex selections, combine the above with the various navigation methods on range objects listed in the first table on this page.

Another method to select a portion of the document is to use the position properties of the range object. Two positioning systems are available and can be combined arbitrarily:

- **Absolute** text cursor positions, starting with position 0 at the document beginning, can be set and retrieved for the beginning and end of a range. For more information see FirstTextPosition 1164 and LastTextPosition 1175. This method requires complex internal calculations and should be used with care.

- The **XMLData** element and a text position inside this element, can be set and retrieved for the beginning and end of a range. For more information see FirstXMLData 1164, FirstXMLDataOffset 1165, LastXMLData 1176, and LastXMLDataOffset 1177. This method is very efficient but requires knowledge of the underlying document structure. It can be used to locate XMLData objects and perform operations on them otherwise not accessible through the user interface.

Modifications to the document content can be achieved by various methods:

- The Text 1184 property allows you to retrieve the document text selected by the range object. If set, the selected document text gets replaced with the new text.
- The standard document edit functions Cut 1161, Copy 1161, Paste 1179 and Delete 1161.
- Table operations for tables that can grow dynamically.
- Methods that map the functionality of the Authentic entry helper windows.
- Access to the XMLData 1309 objects of the underlying document to modify them directly.

25.3.2.5.1 AppendRow

Method: AppendRow()as Boolean

Description

If the beginning of the range is inside a dynamic table, this method inserts a new row at the end of the selected table. The selection of the range is modified to point to the beginning of the new row. The function returns *true* if the append operation was successful, otherwise *false*.

Errors
2001 The authentic range object or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

Examples

```
' -------------------------------------------------------
' Scripting environment - VBScript
' Append row at end of current dynamically growable table
' -------------------------------------------------------
Dim objRange
' we assume that the active document is open in authentic view mode
Set objRange = Application.ActiveDocument.AuthenticView.Selection

' check if we can insert something
If objRange.IsInDynamicTable Then
        objRange.AppendRow
        ' objRange points to beginning of new row
        objRange.Select
End If
```

25.3.2.5.2 Application

Property: Application as Application 1131 (read-only)

Description
Accesses the XMLSpy application object.

Errors
2001 The authentic range object or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

25.3.2.5.3 CanPerformAction

Method: CanPerformAction (*eAction* as SPYAuthenticActions 1359, *strElementName* as String) as Boolean

Description
CanPerformAction and its related methods enable access to the entry-helper functions of Authentic. This function allows easy and consistent modification of the document content, without having to know exactly where the modification will take place. The beginning of the range object is used to locate the next valid location where the specified action can be performed. If the location can be found, the method returns *True*, otherwise it returns *False*.

HINT: To find out all valid element names for a given action, use CanPerformActionWith 1160.

Errors
2001 The authentic range object or its related view object is no longer valid.

2005 Invalid address for the return parameter was specified.
2007 Invalid action was specified.

Examples
See PerformAction 1180.

25.3.2.5.4 CanPerformActionWith

Method: CanPerformActionWith (*eAction* as SPYAuthenticActions 1359, *out_arrElementNames* as Variant)

Description
PerformActionWith and its related methods, enable access to the entry-helper functions of Authentic. This function allows easy and consistent modification of the document content without having to know exactly where the modification will take place.

This method returns an array of those element names that the specified action can be performed with.

HINT: To apply the action use CanPerformActionWith 1160.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.
2007 Invalid action was specified.

Examples
See PerformAction 1180.

25.3.2.5.5 Clone

Method: Clone() as AuthenticRange 1157

Description
Returns a copy of the range object.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

25.3.2.5.6 CollapsToBegin

Method: CollapsToBegin() as AuthenticRange 1157

Description
Sets the end of the range object to its begin. The method returns the modified range object.

Errors
2001 The authentic range object, or its related view object is no longer valid.

(C) 2015-2021 Altova GmbH

2005 Invalid address for the return parameter was specified.

25.3.2.5.7 CollapsToEnd

Method: CollapsToEnd() as AuthenticRange 1157

Description
Sets the beginning of the range object to its end. The method returns the modified range object.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

25.3.2.5.8 Copy

Method: Copy() as Boolean

Description
Returns *False* if the range contains no portions of the document that may be copied.
Returns *True* if text, and in case of fully selected XML elements the elements as well, has been copied to the copy/paste buffer.

Errors
2001 The authentic range object or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

25.3.2.5.9 Cut

Method: Cut() as Boolean

Description
Returns *False* if the range contains portions of the document that may not be deleted.
Returns *True* after text, and in case of fully selected XML elements the elements as well, has been deleted from the document and saved in the copy/paste buffer.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

25.3.2.5.10 Delete

Method: Delete() as Boolean

Description
Returns *False* if the range contains portions of the document that may not be deleted.
Returns *True* after text, and in case of fully selected XML elements the elements as well, has been deleted from the document.

Errors
2001 The authentic range object or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

25.3.2.5.11 DeleteRow

Method: DeleteRow() as Boolean

Description
If the beginning of the range is inside a dynamic table, this method deletes the selected row. The selection of the range gets modified to point to the next element after the deleted row. The function returns *true,* if the delete operation was successful, otherwise *false*.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

Examples

```
' -------------------------------------------------------
' Scripting environment - VBScript
' Delete selected row from dynamically growing table
' -------------------------------------------------------
Dim objRange
' we assume that the active document is open in authentic view mode
Set objRange = Application.ActiveDocument.AuthenticView.Selection

' check if we are in a table
If objRange.IsInDynamicTable Then
        objRange.DeleteRow
End If
```

25.3.2.5.12 DuplicateRow

Method: DuplicateRow() as Boolean

Description
If the beginning of the range is inside a dynamic table, this method inserts a duplicate of the current row after the selected one. The selection of the range gets modified to point to the beginning of the new row. The function returns *true* if the duplicate operation was successful, otherwise *false*.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

(C) 2015-2021 Altova GmbH

Examples

```
' -------------------------------------------------------
' Scripting environment - VBScript
' duplicate row in current dynamically growable table
' -------------------------------------------------------
Dim objRange
' we assume that the active document is open in authentic view mode
Set objRange = Application.ActiveDocument.AuthenticView.Selection

' check if we can insert something
If objRange.IsInDynamicTable Then
	objRange.DuplicateRow
	' objRange points to beginning of new row
	objRange.Select
End If
```

25.3.2.5.13 EvaluateXPath

Method: EvaluateXPath (strExpression as string) strValue as string

Return Value
The method returns a string

Description
`EvaluateXPath()` executes an XPath expression with the context node being the beginning of the range selection. The result is returned as a string, in the case of a sequence it is a space-separated string. If XML context node is irrelevant, the user may provide any node, like `AuthenticView.XMLDataRoot`.

Errors

2001	Invalid object
2005	Invalid parameter
2008	Internal error
2202	Missing context node
2211	XPath error

25.3.2.5.14 ExpandTo

Method: ExpandTo (*eKind* as SPYAuthenticElementKind[1360]), as AuthenticRange[1157]

Description
Selects the whole element of type eKind, that starts at, or contains, the first cursor position of the range. The method returns the modified range object.

Errors

2001	The authentic range object, or its related view object is no longer valid.
2003	Range expansion would be beyond end of document.
2005	Invalid address for the return parameter was specified.

25.3.2.5.15 FirstTextPosition

Property: FirstTextPosition as Long

Description
Set or get the left-most text position index of the range object. This index is always less or equal to LastTextPosition[1175]. Indexing starts with 0 at document beginning, and increments with every different position that the text cursor can occupy. Incrementing the test position by 1, has the same effect as the cursor-right key. Decrementing the test position by 1 has the same effect as the cursor-left key.

If you set FirstTextPosition to a value greater than the current LastTextPosition[1175], LastTextPosition[1175] gets set to the new FirstTextPosition.

HINT: Use text cursor positions with care, since this is a costly operation compared to XMLData based cursor positioning.

Errors

2001	The authentic range object, or its related view object is not valid.
2005	Invalid address for the return parameter was specified.
2006	A text position outside the document was specified.

Examples

```
' ----------------------------------------
' Scripting environment - VBScript
' ----------------------------------------
Dim objAuthenticView
' we assume that the active document is open in authentic view mode
Set objAuthenticView = Application.ActiveDocument.AuthenticView

nDocStartPosition = objAuthenticView.DocumentBegin.FirstTextPosition
nDocEndPosition = objAuthenticView.DocumentEnd.FirstTextPosition

' let's create a range that selects the whole document
' in an inefficient way
Dim objRange
' we need to get a (any) range object first
Set objRange = objAuthenticView.DocumentBegin
objRange.FirstTextPosition = nDocStartPosition
objRange.LastTextPosition = nDocEndPosition

' let's check if we got it right
If objRange.isEqual(objAuthenticView.WholeDocument) Then
	MsgBox "Test using direct text cursor positioning was ok"
Else
	MsgBox "Ooops!"
End If
```

25.3.2.5.16 FirstXMLData

Property: FirstXMLData as XMLData[1309]

Description
Set or get the first XMLData element in the underlying document that is partially, or completely selected by the range. The exact beginning of the selection is defined by the FirstXMLDataOffset[1165] attribute.

Whenever you set FirstXMLData to a new data object, FirstXMLDataOffset[1165] gets set to the first cursor position inside this element. Only XMLData objects that have a cursor position may be used. If you set FirstXMLData / FirstXMLDataOffset[1165] selects a position greater then the current LastXMLData[1176] / LastXMLDataOffset[1177], the latter gets moved to the new start position.

HINT: You can use the FirstXMLData[1164] and LastXMLData[1176] properties to directly access and manipulate the underlying XML document in those cases where the methods available with the AuthenticRange[1157] object are not sufficient.

Errors

2001	The authentic range object, or its related view object is not valid.
2005	Invalid address for the return parameter was specified.
2008	Internal error
2009	The XMLData object cannot be accessed.

Examples

```
' ---------------------------------------------
' Scripting environment - VBScript
' show name of currently selected XMLData element
' ---------------------------------------------
Dim objAuthenticView
' we assume that the active document is open in authentic view mode
Set objAuthenticView = Application.ActiveDocument.AuthenticView

Dim objXmlData
Set objXMLData = objAuthenticView.Selection.FirstXMLData
' authentic view adds a 'text' child element to elements
' of the document which have content. So we have to go one
' element up.
Set objXMLData = objXMLData.Parent
MsgBox "Current selection selects element " & objXMLData.Name
```

25.3.2.5.17 FirstXMLDataOffset

Property: FirstXMLDataOffset as Long

Description
Set or get the cursor position offset inside FirstXMLData[1164] element for the beginning of the range. Offset positions are based on the characters returned by the Text[1184] property, and start with 0. When setting a new offset, use -1 to set the offset to the last possible position in the element. The following cases require specific attention:

- The textual form of entries in Combo Boxes, Check Boxes and similar controls can be different from what you see on screen. Although the data offset is based on this text, there only two valid offset positions, one at the beginning and one at the end of the entry. An attempt to set the offset to somewhere in the middle of the entry, will result in the offset being set to the end.

- The textual form of XML Entities might differ in length from their representation on the screen. The offset is based on this textual form.

If FirstXMLData / FirstXMLDataOffset[1165] selects a position after the current LastXMLData[1176] / LastXMLDataOffset[1177], the latter gets moved to the new start position.

Errors

2001 The authentic range object, or its related view object is not valid.
2005 Invalid offset was specified.
Invalid address for the return parameter was specified.

Examples

```
' ---------------------------------------------
' Scripting environment - VBScript
' Select the complete text of an XMLData element
' using XMLData based selection and ExpandTo
' ---------------------------------------------
Dim objAuthenticView
' we assume that the active document is open in authentic view mode
Set objAuthenticView = Application.ActiveDocument.AuthenticView

' first we use the XMLData based range properties
' to select all text of the first XMLData element
' in the current selection
Dim objRange
Set objRange = objAuthenticView.Selection
objRange.FirstXMLDataOffset = 0  ' start at beginning of element text
objRange.LastXMLData = objRange.FirstXMLData  ' select only one element
objRange.LastXMLDataOffset = -1  ' select till its end

' the same can be achieved with the ExpandTo method
Dim objRange2
Set objRange2 = objAuthenticView.Selection.ExpandTo(spyAuthenticTag)

' were we successful?
If objRange.IsEqual(objRange2) Then
	objRange.Select()
Else
	MsgBox "Oops"
End If
```

25.3.2.5.18 GetElementAttributeNames

Method: GetElementAttributeNames (*strElementName* as String, *out_arrAttributeNames* as Variant)

Description

Retrieve the names of all attributes for the enclosing element with the specified name. Use the element/attribute pairs, to set or get the attribute value with the methods GetElementAttributeValue[1167] and SetElementAttributeValue[1183].

Errors

2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid element name was specified.
Invalid address for the return parameter was specified.

Examples
See SetElementAttributeValue 1183.

25.3.2.5.19 GetElementAttributeValue

Method: GetElementAttributeValue (*strElementName* as String, *strAttributeName* as String) as String

Description
Retrieve the value of the attribute specified in strAttributeName, for the element identified with strElementName. If the attribute is supported but has no value assigned, the empty string is returned. To find out the names of attributes supported by an element, use GetElementAttributeNames 1166, or HasElementAttribute 1171.

Errors

2001	The authentic range object, or its related view object is no longer valid.
2005	Invalid element name was specified. Invalid attribute name was specified. Invalid address for the return parameter was specified.

Examples
See SetElementAttributeValue 1183.

25.3.2.5.20 GetElementHierarchy

Method: GetElementHierarchy (*out_arrElementNames* as Variant)

Description
Retrieve the names of all XML elements that are parents of the current selection. Inner elements get listed before enclosing elements. An empty list is returned whenever the current selection is not inside a single XMLData element.

The names of the element hierarchy, together with the range object uniquely identify XMLData elements in the document. The attributes of these elements can be directly accessed by GetElementAttributeNames 1166, and related methods.

Errors

2001	The authentic range object, or its related view object is no longer valid.
2005	Invalid address for the return parameter was specified.

C# Examples

```
' ---------------------------------------------
' C#
' ---------------------------------------------

namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
```

```
        {
            XMLSpyLib.Application app = new XMLSpyLib.Application();

            app.ShowApplication(true);

            XMLSpyLib.AuthenticView view = app.ActiveDocument.AuthenticView;
            XMLSpyLib.AuthenticRange range = view.DocumentBegin;

            object o = null;
            range.GetElementHierarchy(ref o);

            object[] elements = (object[])o;

            foreach (string e in elements)
            {
                Console.WriteLine(e);
            }
        }
    }
}
```

Also see: SetElementAttributeValue 1183.

25.3.2.5.21 GetEntityNames

Method: GetEntityNames (*out_arrEntityNames* as Variant)

Description
Retrieve the names of all defined entities. The list of retrieved entities is independent of the current selection, or location. Use one of these names with the InsertEntity 1171 function.

Errors

2001	The authentic range object, or its related view object is no longer valid.
2005	Invalid address for the return parameter was specified.

Examples
See: GetElementHierarchy 1167 and InsertEntity 1171.

25.3.2.5.22 GetVariableValue

Method: GetVariableValue `(strName as string) strVal as string`

Return Value
Gets the value of the variable named as the method's parameter.

Errors

2001	Invalid object.
2202	No context.
2204	No such variable in scope
2205	Variable cannot be evaluated
2206	Variable returns sequence

2209 Invalid parameter

25.3.2.5.23 Goto

Method: Goto (*eKind* as SPYAuthenticElementKind[1360], *nCount* as Long, *eFrom* as SPYAuthenticDocumentPosition[1359]) as AuthenticRange[1157]

Description
Sets the range to point to the beginning of the nCount element of type eKind. The start position is defined by the parameter eFrom.

Use positive values for nCount to navigate to the document end. Use negative values to navigate to the beginning of the document. The method returns the modified range object.

Errors

2001	The authentic range object, or its related view object is no longer valid.
2003	Target lies after end of document.
2004	Target lies before begin of document.
2005	Invalid element kind specified. Invalid start position specified. Invalid address for the return parameter was specified.

25.3.2.5.24 GotoNext

Method: GotoNext (*eKind* as SPYAuthenticElementKind[1360]) as AuthenticRange[1157]

Description
Sets the range to the beginning of the next element of type eKind. The method returns the modified range object.

Errors

2001	The authentic range object, or its related view object is no longer valid.
2003	Target lies after end of document.
2005	Invalid element kind specified. Invalid address for the return parameter was specified.

Examples

```
' ---------------------------------------------
' Scripting environment - VBScript
' Scan through the whole document word-by-word
' ---------------------------------------------
Dim objAuthenticView
' we assume that the active document is open in authentic view mode
Set objAuthenticView = Application.ActiveDocument.AuthenticView

Dim objRange
Set objRange = objAuthenticView.DocumentBegin
Dim bEndOfDocument
```

```
bEndOfDocument = False

On Error Resume Next
While Not bEndOfDocument
	objRange.GotoNext(spyAuthenticWord).Select
	If ((Err.number - vbObjecterror) = 2003) Then
		bEndOfDocument = True
		Err.Clear
	ElseIf (Err.number <> 0) Then
		Err.Raise  ' forward error
	End If
Wend
```

25.3.2.5.25 GotoNextCursorPosition

Method: GotoNextCursorPosition() as AuthenticRange 1157

Description
Sets the range to the next cursor position after its current end position. Returns the modified object.

Errors

2001	The authentic range object, or its related view object is no longer valid.
2003	Target lies after end of document.
2005	Invalid address for the return parameter was specified.

25.3.2.5.26 GotoPrevious

Method: GotoPrevious (*eKind* as SPYAuthenticElementKind 1360) as AuthenticRange 1157

Description
Sets the range to the beginning of the element of type eKind which is before the beginning of the current range. The method returns the modified range object.

Errors

2001	The authentic range object, or its related view object is no longer valid.
2004	Target lies before beginning of document.
2005	Invalid element kind specified.
	Invalid address for the return parameter was specified.

Examples

```
' ---------------------------------------------
' Scripting environment - VBScript
' Scan through the whole document tag-by-tag
' ---------------------------------------------
Dim objAuthenticView
' we assume that the active document is open in authentic view mode
Set objAuthenticView = Application.ActiveDocument.AuthenticView

Dim objRange
Set objRange = objAuthenticView.DocumentEnd
```

```
Dim bBeginOfDocument
bBeginOfDocument = False

On Error Resume Next
While Not bBeginOfDocument
	objRange.GotoPrevious(spyAuthenticTag).Select
	If ((Err.number - vbObjecterror) = 2004) Then
		bBeginOfDocument = True
		Err.Clear
	ElseIf (Err.number <> 0) Then
		Err.Raise  ' forward error
	End If
Wend
```

25.3.2.5.27 GotoPreviousCursorPosition

Method: GotoPreviousCursorPosition() as AuthenticRange 1157

Description
Set the range to the cursor position immediately before the current position. Returns the modified object.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2004 Target lies before begin of document.
2005 Invalid address for the return parameter was specified.

25.3.2.5.28 HasElementAttribute

Method: HasElementAttribute (*strElementName* as String, *strAttributeName* as String) as Boolean

Description
Tests if the enclosing element with name strElementName, supports the attribute specified in strAttributeName.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid element name was specified.
Invalid address for the return parameter was specified.

25.3.2.5.29 InsertEntity

Method: InsertEntity (*strEntityName* as String)

Description
Replace the ranges selection with the specified entity. The specified entity must be one of the entity names returned by GetEntityNames 1168.

Errors
2001 The authentic range object, or its related view object is no longer valid.

2005 Unknown entry name was specified.

Examples

```
' ---------------------------------------------------------
' Scripting environment - VBScript
' Insert the first entity in the list of available entities
' ---------------------------------------------------------
Dim objRange
' we assume that the active document is open in authentic view mode
Set objRange = Application.ActiveDocument.AuthenticView.Selection

' first we get the names of all available entities as they
' are shown in the entry helper of XMLSpy
Dim arrEntities
objRange.GetEntityNames arrEntities

' we insert the first one of the list
If UBound(arrEntities) >= 0 Then
	objRange.InsertEntity arrEntities(0)
Else
	MsgBox "Sorry, no entities are available for this document"
End If
```

25.3.2.5.30 InsertRow

Method: InsertRow() as Boolean

Description

If the beginning of the range is inside a dynamic table, this method inserts a new row before the current one. The selection of the range gets modified to point to the beginning of the newly inserted row. The function returns *true* if the insert operation was successful, otherwise *false*.

Errors

2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

Examples

```
' ------------------------------------------------------------
' Scripting environment - VBScript
' Insert row at beginning of current dynamically growing table
' ------------------------------------------------------------
Dim objRange
' we assume that the active document is open in authentic view mode
Set objRange = Application.ActiveDocument.AuthenticView.Selection

' check if we can insert something
If objRange.IsInDynamicTable Then
	objRange.InsertRow
	' objRange points to beginning of new row
	objRange.Select
 End If
```

25.3.2.5.31 IsCopyEnabled

Property: IsCopyEnabled as Boolean (read-only)

Description
Checks if the copy operation is supported for this range.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

25.3.2.5.32 IsCutEnabled

Property: IsCutEnabled as Boolean (read-only)

Description
Checks if the cut operation is supported for this range.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

25.3.2.5.33 IsDeleteEnabled

Property: IsDeleteEnabled as Boolean (read-only)

Description
Checks if the delete operation is supported for this range.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

25.3.2.5.34 IsEmpty

Method: IsEmpty() as Boolean

Description
Tests if the first and last position of the range are equal.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

25.3.2.5.35 IsEqual

Method: IsEqual (*objCmpRange* as AuthenticRange[1157]) as Boolean

Description
Tests if the start and end of both ranges are the same.

Errors
2001 One of the two range objects being compared, is invalid.
2005 Invalid address for a return parameter was specified.

25.3.2.5.36 IsFirstRow

Property: IsFirstRow as Boolean (read-only)

Description
Test if the range is in the first row of a table. Which table is taken into consideration depends on the extent of the range. If the selection exceeds a single row of a table, the check is if this table is the first element in an embedding table. See the entry helpers of the user manual for more information.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

25.3.2.5.37 IsInDynamicTable

Method: IsInDynamicTable() as Boolean

Description
Test if the whole range is inside a table that supports the different row operations like 'insert', 'append', duplicate, etc.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

25.3.2.5.38 IsLastRow

Property: IsLastRow as Boolean (read-only)

Description
Test if the range is in the last row of a table. Which table is taken into consideration depends on the extent of the range. If the selection exceeds a single row of a table, the check is if this table is the last element in an embedding table. See the entry helpers of the user manual for more information.

Errors

(C) 2015-2021 Altova GmbH

2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

25.3.2.5.39 IsPasteEnabled

Property: IsPasteEnabled as Boolean (read-only)

Description
Checks if the paste operation is supported for this range.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

25.3.2.5.40 IsSelected

Property: IsSelected as Boolean

Description
Returns true() if selection is present. The selection range still can be empty: that happens when e.g. only the cursor is set.

25.3.2.5.41 IsTextStateApplied

Method: IsTextStateApplied (*i_strElementName* as String) as Boolean

Description
Checks if all the selected text is embedded into an XML Element with name i_strElementName. Common examples for the parameter i_strElementName are "strong", "bold" or "italic".

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

25.3.2.5.42 LastTextPosition

Property: LastTextPosition as Long

Description
Set or get the rightmost text position index of the range object. This index is always greater or equal to FirstTextPosition 1164. Indexing starts with 0 at the document beginning, and increments with every different position that the text cursor can occupy. Incrementing the test position by 1, has the same effect as the cursor-right key. Decreasing the test position by 1 has the same effect as the cursor-left key.

If you set LastTextPosition to a value less then the current FirstTextPosition[1164], FirstTextPosition[1164] gets set to the new LastTextPosition.

HINT: Use text cursor positions with care, since this is a costly operation compared to XMLData based cursor positioning.

Errors

2001 The authentic range object, or its related view object is not valid.
2005 Invalid address for the return parameter was specified.
2006 A text position outside the document was specified.

Examples

```
' ----------------------------------------
' Scripting environment - VBScript
' ----------------------------------------
Dim objAuthenticView
' we assume that the active document is open in authentic view mode
Set objAuthenticView = Application.ActiveDocument.AuthenticView

nDocStartPosition = objAuthenticView.DocumentBegin.FirstTextPosition
nDocEndPosition = objAuthenticView.DocumentEnd.FirstTextPosition

' let's create a range that selects the whole document
' in an inefficient way
Dim objRange
' we need to get a (any) range object first
Set objRange = objAuthenticView.DocumentBegin
objRange.FirstTextPosition = nDocStartPosition
objRange.LastTextPosition = nDocEndPosition

' let's check if we got it right
If objRange.isEqual(objAuthenticView.WholeDocument) Then
	MsgBox "Test using direct text cursor positioning was ok"
Else
	MsgBox "Oops!"
End If
```

25.3.2.5.43 LastXMLData

Property: LastXMLData as XMLData[1309]

Description

Set or get the last XMLData element in the underlying document that is partially or completely selected by the range. The exact end of the selection is defined by the LastXMLDataOffset[1177] attribute.

Whenever you set LastXMLData to a new data object, LastXMLDataOffset[1177] gets set to the last cursor position inside this element. Only XMLData objects that have a cursor position may be used. If you set LastXMLData / LastXMLDataOffset[1177], select a position less then the current FirstXMLData[1164] / FirstXMLDataOffset[1165], the latter gets moved to the new end position.

HINT: You can use the FirstXMLData[1164] and LastXMLData[1176] properties to directly access and manipulate the underlying XML document in those cases, where the methods available with the AuthenticRange[1157] object are not sufficient.

Errors

2001 The authentic range object, or its related view object is not valid.
2005 Invalid address for the return parameter was specified.
2008 Internal error
2009 The XMLData object cannot be accessed.

25.3.2.5.44 LastXMLDataOffset

Property: LastXMLDataOffset as Long

Description

Set or get the cursor position inside LastXMLData[1176] element for the end of the range.

Offset positions are based on the characters returned by the Text[1184] property and start with 0. When setting a new offset, use -1 to set the offset to the last possible position in the element. The following cases require specific attention:

- The textual form of entries in Combo Boxes, Check Boxes and similar controls can be different from what you see on the screen. Although, the data offset is based on this text, there only two valid offset positions, one at the beginning and one at the end of the entry. An attempt to set the offset to somewhere in the middle of the entry, will result in the offset being set to the end.
- The textual form of XML Entities might differ in length from their representation on the screen. The offset is based on this textual form.

If LastXMLData[1176] / LastXMLDataOffset[1177] selects a position before FirstXMLData[1164] / FirstXMLDataOffset[1165], the latter gets moved to the new end position.

Errors

2001 The authentic range object, or its related view object is not valid.
2005 Invalid offset was specified.
Invalid address for the return parameter was specified.

Examples

```
' ---------------------------------------------
' Scripting environment - VBScript
' Select the complete text of an XMLData element
' using XMLData based selection and ExpandTo
' ---------------------------------------------
Dim objAuthenticView
' we assume that the active document is open in authentic view mode
Set objAuthenticView = Application.ActiveDocument.AuthenticView

' first we use the XMLData based range properties
' to select all text of the first XMLData element
' in the current selection
Dim objRange
Set objRange = objAuthenticView.Selection
objRange.FirstXMLDataOffset = 0  ' start at beginning of element text
```

```
objRange.LastXMLData = objRange.FirstXMLData  ' select only one element
objRange.LastXMLDataOffset = -1  ' select till its end

' the same can be achieved with the ExpandTo method
Dim objRange2
Set objRange2 = objAuthenticView.Selection.ExpandTo(spyAuthenticTag)

' were we successful?
If objRange.IsEqual(objRange2) Then
        objRange.Select()
Else
        MsgBox "Ooops"
End If
```

25.3.2.5.45 MoveBegin

Method: MoveBegin (*eKind* as SPYAuthenticElementKind[1360], *nCount* as Long) as AuthenticRange[1157]

Description
Move the beginning of the range to the beginning of the nCount element of type eKind. Counting starts at the current beginning of the range object.

Use positive numbers for nCount to move towards the document end, use negative numbers to move towards document beginning. The end of the range stays unmoved, unless the new beginning would be larger than it. In this case, the end is moved to the new beginning. The method returns the modified range object.

Errors

2001	The authentic range object, or its related view object is no longer valid.
2003	Target lies after end of document.
2004	Target lies before beginning of document.
2005	Invalid element kind specified.
	Invalid address for the return parameter was specified.

25.3.2.5.46 MoveEnd

Method: MoveEnd (*eKind* as SPYAuthenticElementKind[1360], *nCount* as Long) as AuthenticRange[1157]

Description
Move the end of the range to the begin of the nCount element of type eKind. Counting starts at the current end of the range object.

Use positive numbers for nCount to move towards the document end, use negative numbers to move towards document beginning. The beginning of the range stays unmoved, unless the new end would be less than it. In this case, the beginning gets moved to the new end. The method returns the modified range object.

Errors

2001	The authentic range object, or its related view object is no longer valid.
2003	Target lies after end of document.
2004	Target lies before begin of document.
2005	Invalid element kind specified.
	Invalid address for the return parameter was specified.

25.3.2.5.47 MoveRowDown

Method: MoveRowDown() as Boolean

Description
If the beginning of the range is inside a dynamic table and selects a row which is not the last row in this table, this method swaps this row with the row immediately below. The selection of the range moves with the row, but does not otherwise change. The function returns *true* if the move operation was successful, otherwise *false*.

Errors
2001 The authentic range object or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

25.3.2.5.48 MoveRowUp

Method: MoveRowUp() as Boolean

Description
If the beginning of the range is inside a dynamic table and selects a row which is not the first row in this table, this method swaps this row with the row above. The selection of the range moves with the row, but does not change otherwise. The function returns *true* if the move operation was successful, otherwise *false*.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

25.3.2.5.49 Parent

Property: Parent as AuthenticView[1185] (read-only)

Description
Access the view that owns this range object.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

25.3.2.5.50 Paste

Method: Paste() as Boolean

Description
Returns *False* if the copy/paste buffer is empty, or its content cannot replace the current selection.

Otherwise, deletes the current selection, inserts the content of the copy/paste buffer, and returns *True*.

Errors

2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

25.3.2.5.51 PerformAction

Method: PerformAction (*eAction* as SPYAuthenticActions 1359, *strElementName* as String) as Boolean

Description

PerformAction and its related methods, give access to the entry-helper functions of Authentic. This function allows easy and consistent modification of the document content without a need to know exactly where the modification will take place. The beginning of the range object is used to locate the next valid location where the specified action can be performed. If no such location can be found, the method returns *False*. Otherwise, the document gets modified and the range points to the beginning of the modification.

HINT: To find out element names that can be passed as the second parameter use CanPerformActionWith 1160.

Errors

2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.
2007 Invalid action was specified.

Examples

```
' ---------------------------------------------
' Scripting environment - VBScript
' Insert the innermost element
' ---------------------------------------------
Dim objRange
' we assume that the active document is open in authentic view mode
Set objRange = Application.ActiveDocument.AuthenticView.Selection

' we determine the elements that can be inserted at the current position
Dim arrElements()
objRange.CanPerformActionWith spyAuthenticInsertBefore, arrElements

' we insert the first (innermost) element
If UBound(arrElements) >= 0 Then
	objRange.PerformAction spyAuthenticInsertBefore, arrElements(0)
	' objRange now points to the beginning of the inserted element
	' we set a default value and position at its end
	objRange.Text = "Hello"
	objRange.ExpandTo(spyAuthenticTag).CollapsToEnd().Select
Else
	MsgBox "Can't insert any elements at current position"
End If
```

25.3.2.5.52 Select

Method: Select()

Description
Makes this range the current user interface selection. You can achieve the same result using:
'*objRange.Parent.Selection = objRange*'

Errors
2001 The authentic range object or its related view object is no longer valid.

Examples
```
' ----------------------------------------
' Scripting environment - VBScript
' ----------------------------------------
Dim objAuthenticView
' we assume that the active document is open in authentic view mode
Set objAuthenticView = Application.ActiveDocument.AuthenticView

' set current selection to end of document
objAuthenticView.DocumentEnd.Select()
```

25.3.2.5.53 SelectNext

Method: SelectNext (*eKind* as SPYAuthenticElementKind[1360]) as AuthenticRange[1157]

Description
Selects the element of type eKind after the current end of the range. The method returns the modified range object.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2003 Target lies after end of document.
2005 Invalid element kind specified.
Invalid address for the return parameter was specified.

Examples
```
' --------------------------------------------
' Scripting environment - VBScript
' Scan through the whole document word-by-word
' --------------------------------------------
Dim objAuthenticView
' we assume that the active document is open in authentic view mode
Set objAuthenticView = Application.ActiveDocument.AuthenticView

Dim objRange
Set objRange = objAuthenticView.DocumentBegin
Dim bEndOfDocument
bEndOfDocument = False
```

```
On Error Resume Next
While Not bEndOfDocument
	objRange.SelectNext(spyAuthenticWord).Select
	If ((Err.number - vbObjecterror) = 2003) Then
		bEndOfDocument = True
		Err.Clear
	ElseIf (Err.number <> 0) Then
		Err.Raise  ' forward error
	End If
Wend
```

25.3.2.5.54 SelectPrevious

Method: GotoPrevious (*eKind* as SPYAuthenticElementKind[1360]) as AuthenticRange[1157]

Description
Selects the element of type eKind before the current beginning of the range. The method returns the modified range object.

Errors

2001	The authentic range object, or its related view object is no longer valid.
2004	Target lies before begin of document.
2005	Invalid element kind specified.
	Invalid address for the return parameter was specified.

Examples

```
' ---------------------------------------------
' Scripting environment - VBScript
' Scan through the whole document tag-by-tag
' ---------------------------------------------
Dim objAuthenticView
' we assume that the active document is open in authentic view mode
Set objAuthenticView = Application.ActiveDocument.AuthenticView

Dim objRange
Set objRange = objAuthenticView.DocumentEnd
Dim bBeginOfDocument
bBeginOfDocument = False

On Error Resume Next
While Not bBeginOfDocument
	objRange.SelectPrevious(spyAuthenticTag).Select
	If ((Err.number - vbObjecterror) = 2004) Then
		bBeginOfDocument = True
		Err.Clear
	ElseIf (Err.number <> 0) Then
		Err.Raise  ' forward error
	End If
Wend
```

25.3.2.5.55 SetElementAttributeValue

Method: SetElementAttributeValue (*strElementName* as String, *strAttributeName* as String, *strAttributeValue* as String)

Description
Set the value of the attribute specified in strAttributeName for the element identified with strElementName. If the attribute is supported but has no value assigned, the empty string is returned. To find out the names of attributes supported by an element, use GetElementAttributeNames 1166, or HasElementAttribute 1171.

Errors

2001	The authentic range object or its related view object is no longer valid.
2005	Invalid element name was specified. Invalid attribute name was specified. Invalid attribute value was specified.

Examples

```
' ---------------------------------------------
' Scripting environment - VBScript
' Get and set element attributes
' ---------------------------------------------
Dim objRange
' we assume that the active document is open in authentic view mode
Set objRange = Application.ActiveDocument.AuthenticView.Selection

' first we find out all the elements below the beginning of the range
Dim arrElements
objRange.GetElementHierarchy arrElements

If IsArray(arrElements) Then
	If UBound(arrElements) >= 0 Then
		' we use the top level element and find out its valid attributes
		Dim arrAttrs()
		objRange.GetElementAttributeNames arrElements(0), arrAttrs

		If UBound(arrAttrs) >= 0 Then
			' we retrieve the current value of the first valid attribute
			Dim strAttrVal
			strAttrVal = objRange.GetElementAttributeValue (arrElements(0), arrAttrs(0))
			msgbox "current value of " & arrElements(0) & "//" & arrAttrs(0) & " is: " & strAttrVal

			' we change this value and read it again
			strAttrVal = "Hello"
			objRange.SetElementAttributeValue arrElements(0), arrAttrs(0), strAttrVal
			strAttrVal = objRange.GetElementAttributeValue (arrElements(0), arrAttrs(0))
			msgbox "new value of " & arrElements(0) & "//" & arrAttrs(0) & " is: " & strAttrVal
		End If
	End If
End If
```

25.3.2.5.56 SetFromRange

Method: SetFromRange (*objSrcRange* as AuthenticRange[1157])

Description
Sets the range object to the same beginning and end positions as objSrcRange.

Errors
2001 One of the two range objects, is invalid.
2005 Null object was specified as source object.

25.3.2.5.57 SetVariableValue

Method: SetVariableValue`(strName as string, strValue as string)`

Return Value
Sets the value (second parameter) of the variable named in the first parameter.

Errors
2201 Invalid object.
2202 No context.
2204 No such variable in scope
2205 Variable cannot be evaluated
2206 Variable returns sequence
2207 Variable read-only
2208 No modification allowed

25.3.2.5.58 Text

Property: Text as String

Description
Set or get the textual content selected by the range object.

The number of characters retrieved are not necessarily identical, as there are text cursor positions between the beginning and end of the selected range. Most document elements support an end cursor position different to the beginning cursor position of the following element. Drop-down lists maintain only one cursor position, but can select strings of any length. In the case of radio buttons and check boxes, the text property value holds the string of the corresponding XML element.

If the range selects more then one element, the text is the concatenation of the single texts. XML entities are expanded so that '&' is expected as '&'.

Setting the text to the empty string, does not delete any XML elements. Use Cut[1161], Delete[1161] or PerformAction[1180] instead.

Errors

2001 The authentic range object or its related view object is no longer valid.
2005 Invalid address for a return parameter was specified.

25.3.2.6 AuthenticView

Properties
Application 1194
AsXMLString 1195
DocumentBegin 1196
DocumentEnd 1196
Event 1197
MarkupVisibility 1200
Parent 1200
Selection 1201
XMLDataRoot 1203
WholeDocument 1203

Methods
Goto 1198
IsRedoEnabled 1199
IsUndoEnabled 1199
Print 1200
Redo 1201
Undo 1202
UpdateXMLInstanceEntities 1202

Events
OnBeforeCopy 1186
OnBeforeCut 1186
OnBeforeDelete 1187
OnBeforeDrop 1187
OnBeforePaste 1188
OnDragOver 1189
OnKeyBoardEvent 1190
OnMouseEvent 1191
OnSelectionChanged 1192

Description

AuthenticView and its child objects AuthenticRange 1157 and AuthenticDataTransfer provide you with an interface for Authentic View, which allow easy and consistent modification of document contents. These interfaces replace the following interfaces which are marked now as **obsolete:**

OldAuthenticView (old name was DocEditView)
AuthenticSelection (old name was DocEditSelection, superseded by AuthenticRange 1157)
AuthenticEvent (old name was DocEditEvent)

AuthenticView gives you easy access to specific features such as printing, the multi-level undo buffer, and the current cursor selection, or position.

AuthenticView uses objects of type AuthenticRange 1157 to make navigation inside the document straight-forward, and to allow for the flexible selection of logical text elements. Use the properties DocumentBegin 1196, DocumentEnd 1196, or WholeDocument 1203 for simple selections, while using the Goto 1198 method for more complex selections. To navigate relative to a given document range, see the methods and properties of the AuthenticRange 1157 object.

Examples

```
' ---------------------------------------
' XMLSpy scripting environment - VBScript
' secure access to authentic view object
' ---------------------------------------
Dim objDocument
Set objDocument = Application.ActiveDocument
If (Not objDocument Is Nothing) Then
	' we have an active document, now check for view mode
	If (objDocument.CurrentViewMode <> spyViewAuthentic) Then
		If (Not objDocument.SwitchViewMode (spyViewAuthentic)) Then
			MsgBox "Active document does not support authentic view mode"
		Else
			' now it is safe to access the authentic view object
```

```
                    Dim objAuthenticView
                    Set objAuthenticView = objDocument.AuthenticView
                    ' now use the authentic view object

            End If
        End If
Else
        MsgBox "No document is open"
End If
```

25.3.2.6.1 Events

25.3.2.6.1.1 OnBeforeCopy

Event: OnBeforeCopy() as Boolean

Scripting environment - VBScript:

```
Function On_AuthenticBeforeCopy()
        ' On_AuthenticBeforeCopy = False  ' to disable operation
End Function
```

Scripting environment - JScript:

```
function On_AuthenticBeforeCopy()
{
        // return false;  /* to disable operation */
}
```

IDE Plugin:

IXMLSpyPlugIn.OnEvent (21, ...)// nEventId = 21

Description

This event gets triggered before a copy operation gets performed on the document. Return *True* (or nothing) to allow copy operation. Return *False* to disable copying.

25.3.2.6.1.2 OnBeforeCut

Event: OnBeforeCut() as Boolean

Scripting environment - VBScript:

```
Function On_AuthenticBeforeCut()
        ' On_AuthenticBeforeCut = False  ' to disable operation
End Function
```

Scripting environment - JScript:

```
function On_AuthenticBeforeCut()
{
        // return false;  /* to disable operation */
}
```

IDE Plugin:

IXMLSpyPlugIn.OnEvent (20, ...)// nEventId = 20

Description
This event gets triggered before a cut operation gets performed on the document. Return *True* (or nothing) to allow cut operation. Return *False* to disable operation.

25.3.2.6.1.3 OnBeforeDelete

Event: OnBeforeDelete() as Boolean

Scripting environment - VBScript:
```
Function On_AuthenticBeforeDelete()
        ' On_AuthenticBeforeDelete = False  ' to disable operation
End Function
```

Scripting environment - JScript:
```
function On_AuthenticBeforeDelete()
{
        // return false;  /* to disable operation */
}
```

IDE Plugin:
IXMLSpyPlugIn.OnEvent (22, ...)// nEventId = 22

Description
This event gets triggered before a delete operation gets performed on the document. Return *True* (or nothing) to allow delete operation. Return *False* to disable operation.

25.3.2.6.1.4 OnBeforeDrop

Event: OnBeforeDrop (*i_nXPos* as Long, *i_nYPos* as Long, *i_ipRange* as AuthenticRange 1157, *i_ipData* as cancelBoolean

Scripting environment - VBScript:
```
Function On_AuthenticBeforeDrop(nXPos, nYPos, objRange, objData)
        ' On_AuthenticBeforeDrop = False  ' to disable operation
End Function
```

Scripting environment - JScript:
```
function On_AuthenticBeforeDrop(nXPos, nYPos, objRange, objData)
{
        // return false;  /* to disable operation */
}
```

IDE Plugin:
IXMLSpyPlugIn.OnEvent (11, ...)// nEventId = 11

Description
This event gets triggered whenever a previously dragged object gets dropped inside the application window. All event related information gets passed as parameters.

The first two parameters specify the mouse position at the time when the event occurred. The parameter *objRange* passes a range object that selects the XML element below the mouse position. The value of this parameter might be *NULL*. Be sure to check before you access the range object. The parameter *objData* allows to access information about the object being dragged.

Return *False* to cancel the drop operation. Return *True* (or nothing) to continue normal operation.

Examples

```
' ---------------------------------------------------------------------------
' VB code snippet - connecting to object level events
' ---------------------------------------------------------------------------
' access XMLSpy (without checking for any errors)
Dim objSpy As XMLSpyLib.Application
Set objSpy = GetObject("", "XMLSpy.Application")

' this is the event callback routine connected to the OnBeforeDrop
' event of object objView
Private Function objView_OnBeforeDrop(ByVal i_nXPos As Long, ByVal i_nYPos As Long,
                                      ByVal i_ipRange As IAuthenticRange,
                                      ByVal i_ipData As IAuthenticDataTransfer) As Boolean

   If (Not i_ipRange Is Nothing) Then
      MsgBox ("Dropping on content is prohibited");
      Return False;
   Else
      Return True;
   End If
End Function

' use VBA keyword WithEvents to connect to object-level event
Dim WithEvents objView As XMLSpyLib.AuthenticView
Set objView = objSpy.ActiveDocument.AuthenticView

' continue here with something useful ...
' and serve the windows message loop
```

25.3.2.6.1.5 OnBeforePaste

Event: OnBeforePaste (*objData* as Variant, *strType* as String) as Boolean

Scripting environment - VBScript:

```
Function On_AuthenticBeforePaste(objData, strType)
        ' On_AuthenticBeforePaste = False  ' to disable operation
End Function
```

Scripting environment - JScript:

```
function On_AuthenticBeforePaste(objData, strType)
{
        // return false;  /* to disable operation */
}
```

IDE Plugin:

IXMLSpyPlugIn.OnEvent (19, ...)// nEventId = 19

Description
This event gets triggered before a paste operation gets performed on the document. The parameter *strType* is one of "TEXT", "UNICODETEXT" or "IUNKNOWN". In the first two cases *objData* contains a string representation of the object that will be pasted. In the later case, *objData* contains a pointer to an IUnknown COM interface.

Return *True* (or nothing) to allow paste operation. Return *False* to disable operation.

25.3.2.6.1.6 OnBeforeSave

Event: OnBeforeSave (SaveAs flag) as Boolean

Description: `OnBeforeSave` gives the opportunity to e.g. warn the user about overwriting the existing XML document, or to make the document read-only when specific circumstances are not met. The event will be fired before the file dialog is shown.

25.3.2.6.1.7 OnDragOver

Event: OnDragOver (*nXPos* as Long, *nYPos* as Long, *eMouseEvent* as SPYMouseEvent[1364], *objRange* as AuthenticRange[1157], *objData* as AuthenticDataTransfer) as Boolean

Scripting environment - VBScript:
```
Function On_AuthenticDragOver(nXPos, nYPos, eMouseEvent, objRange, objData)
        ' On_AuthenticDragOver = False  ' to disable operation
End Function
```

Scripting environment - JScript:
```
function On_AuthenticDragOver(nXPos, nYPos, eMouseEvent, objRange, objData)
{
        // return false;  /* to disable operation */
}
```

IDE Plugin:
IXMLSpyPlugIn.OnEvent (10, ...)// nEventId = 10

Description
This event gets triggered whenever an object from within or outside of Authentic View gets dragged with the mouse over the application window. All event related information gets passed as parameters.

The first three parameters specify the mouse position, the mouse button status and the status of the virtual keys at the time when the event occurred. The parameter *objRange* passes a range object that selects the XML element below the mouse position. The value of this parameter might be *NULL*. Be sure to check before you access the range object. The parameter *objData* allows to access information about the object being dragged.

Return *False* to cancel the drag operation. Return *True* (or nothing) to continue normal operation.

Examples
```
' ---------------------------------------------------------------------------
' VB code snippet - connecting to object level events
' ---------------------------------------------------------------------------
```

```
' access XMLSpy (without checking for any errors)
Dim objSpy As XMLSpyLib.Application
Set objSpy = GetObject("", "XMLSpy.Application")

' this is the event callback routine connected to the OnDragOver
' event of object objView
Private Function objView_OnDragOver(ByVal i_nXPos As Long, ByVal i_nYPos As Long,
                                    ByVal i_eMouseEvent As SPYMouseEvent,
                                    ByVal i_ipRange As IAuthenticRange,
                                    ByVal i_ipData As IAuthenticDataTransfer) As Boolean

   If (((i_eMouseEvent And spyShiftKeyDownMask) <> 0) And
          (Not i_ipRange Is Nothing)) Then
      MsgBox ("Floating over element " & i_ipRange.FirstXMLData.Parent.Name);
   End If

   Return True;
End Function

' use VBA keyword WithEvents to connect to object-level event
Dim WithEvents objView As XMLSpyLib.AuthenticView
Set objView = objSpy.ActiveDocument.AuthenticView

' continue here with something useful ...
' and serve the windows message loop
```

25.3.2.6.1.8 OnKeyboardEvent

Event: OnKeyboardEvent (*eKeyEvent* as SPYKeyEvent[1363], *nKeyCode* as Long, *nVirtualKeyStatus* as Long) as Boolean

Scripting environment - VBScript:

```
Function On_AuthenticKeyboardEvent(eKeyEvent, nKeyCode, nVirtualKeyStatus)
        ' On_AuthenticKeyboardEvent = True  ' to cancel bubbling of event
End Function
```

Scripting environment - JScript:

```
function On_AuthenticKeyboardEvent(eKeyEvent, nKeyCode, nVirtualKeyStatus)
{
        // return true;  /* to cancel bubbling of event */
}
```

IDE Plugin:

IXMLSpyPlugIn.OnEvent (30, ...)// nEventId = 30

Description

This event gets triggered for *WM_KEYDOWN*, *WM_KEYUP* and *WM_CHAR* Windows messages.

The actual message type is available in the *eKeyEvent* parameter. The status of virtual keys is combined in the parameter *nVirtualKeyStatus*. Use the bit-masks defined in the enumeration datatype SPYVirtualKeyMask[1371], to test for the different keys or their combinations.

25.3.2.6.1.9 OnLoad

Event: OnLoad ()

Description: `OnLoad` can be used e.g. to restrict some AuthenticView functionality, as shown in the example below:

```
function On_AuthenticLoad(  )
{
	// We are disabling all entry helpers in order to prevent user from manipulating XML tree
	AuthenticView.DisableElementEntryHelper();
	AuthenticView.DisableAttributeEntryHelper();

	// We are also disabling the markup buttons for the same purpose
	AuthenticView.SetToolbarButtonState( 'AuthenticMarkupSmall', authenticToolbarButtonDisabled );
	AuthenticView.SetToolbarButtonState( 'AuthenticMarkupLarge', authenticToolbarButtonDisabled );
	AuthenticView.SetToolbarButtonState( 'AuthenticMarkupMixed', authenticToolbarButtonDisabled );
}
```

In the example the status of the Markup Small, Markup Large, Markup Mixed toolbar buttons are manipulated with the help of button identifiers. See complete list[1192].

25.3.2.6.1.10 OnMouseEvent

Event: OnMouseEvent (*nXPos* as Long, *nYPos* as Long, *eMouseEvent* as SPYMouseEvent[1364], *objRange* as AuthenticRange[1157]) as Boolean

Scripting environment - VBScript:

```
Function On_AuthenticMouseEvent(nXPos, nYPos, eMouseEvent, objRange)
	' On_AuthenticMouseEvent = True  ' to cancel bubbling of event
End Function
```

Scripting environment - JScript:

```
function On_AuthenticMouseEvent(nXPos, nYPos, eMouseEvent, objRange)
{
	// return true;  /* to cancel bubbling of event */
}
```

IDE Plugin:

IXMLSpyPlugIn.OnEvent (31, ...)// nEventId = 31

Description

This event gets triggered for every mouse movement and mouse button Windows message.

The actual message type and the mouse buttons status, is available in the *eMouseEvent* parameter. Use the bit-masks defined in the enumeration datatype SPYMouseEvent[1364] to test for the different messages, button status, and their combinations.

The parameter *objRange* identifies the part of the document found at the current mouse cursor position. The range object always selects a complete tag of the document. (This might change in future versions, when a more precise positioning mechanism becomes available). If no selectable part of the document is found at the current position, the range object is *null*.

25.3.2.6.1.11 OnSelectionChanged

Event: OnSelectionChanged (*objNewSelection* as AuthenticRange[1157])

Scripting environment - VBScript:
Function On_AuthenticSelectionChanged (*objNewSelection*)
End Function

Scripting environment - JScript:
function On_AuthenticSelectionChanged (*objNewSelection*)
{
}

IDE Plugin:
IXMLSpyPlugIn.OnEvent (23, ...)// nEventId = 23

Description
This event gets triggered whenever the selection in the user interface changes.

Examples

```
' ---------------------------------------------------------------------------
' VB code snippet - connecting to object level events
' ---------------------------------------------------------------------------
' access XMLSpy (without checking for any errors)
Dim objSpy As XMLSpyLib.Application
Set objSpy = GetObject("", "XMLSpy.Application")

' this is the event callback routine connected to the OnSelectionChanged
' event of object objView
Private Sub objView_OnSelectionChanged (ByVal i_ipNewRange As XMLSpyLib.IAuthenticRange)
    MsgBox ("new selection: " & i_ipNewRange.Text)
End Sub

' use VBA keyword WithEvents to connect to object-level event
Dim WithEvents objView As XMLSpyLib.AuthenticView
Set objView = objSpy.ActiveDocument.AuthenticView

' continue here with something useful ...
' and serve the windows message loop
```

25.3.2.6.1.12 OnToolbarButtonClicked

Event: OnToolbarButtonClicked (Button identifier)

Description: `OnToolbarButtonClicked` is fired when a toolbar button was clicked by user. The parameter button identifier helps to determine which button was clicked. The list of predefined button identifiers is below:

- AuthenticPrint
- AuthenticPrintPreview
- AuthenticUndo
- AuthenticRedo
- AuthenticCut

- AuthenticCopy
- AuthenticPaste
- AuthenticClear
- AuthenticMarkupHide
- AuthenticMarkupLarge
- AuthenticMarkupMixed
- AuthenticMarkupSmall
- AuthenticValidate
- AuthenticChangeWorkingDBXMLCell
- AuthenticSave
- AuthenticSaveAs
- AuthenticReload
- AuthenticTableInsertRow
- AuthenticTableAppendRow
- AuthenticTableDeleteRow
- AuthenticTableInsertCol
- AuthenticTableAppendCol
- AuthenticTableDeleteCol
- AuthenticTableJoinCellRight
- AuthenticTableJoinCellLeft
- AuthenticTableJoinCellAbove
- AuthenticTableJoinCellBelow
- AuthenticTableSplitCellHorizontally
- AuthenticTableSplitCellVertically
- AuthenticTableAlignCellContentTop
- AuthenticTableCenterCellVertically
- AuthenticTableAlignCellContentBottom
- AuthenticTableAlignCellContentLeft
- AuthenticTableCenterCellContent
- AuthenticTableAlignCellContentRight
- AuthenticTableJustifyCellContent
- AuthenticTableInsertTable
- AuthenticTableDeleteTable
- AuthenticTableProperties
- AuthenticAppendRow
- AuthenticInsertRow
- AuthenticDuplicateRow
- AuthenticMoveRowUp
- AuthenticMoveRowDown
- AuthenticDeleteRow
- AuthenticDefineEntities
- AuthenticXMLSignature

For custom buttons the user might add his own identifiers. Please, note that the user must take care, as the identifiers are not checked for uniqueness. The same identifiers can be used to identify buttons in the `Set/GetToolbarState()` COM API calls. By adding code for different buttons, the user is in the position to completely redefine the AuthenticView toolbar behavior, adding own methods for table manipulation, etc.

25.3.2.6.1.13 OnToolbarButtonExecuted

Event: OnToolbarButtonExecuted (Button identifier)

Description: `OnToolbarButtonClicked` is fired when a toolbar button was clicked by user. The parameter button identifier helps to determine which button was clicked. See the list of predefined button identifiers[1192].

`OnToolbarButtonExecuted` is fired after the toolbar action was executed. It is useful e.g. to add update code, as shown in the example below:

```
//event fired when a toolbar button action was executed
function On_AuthenticToolbarButtonExecuted( varBtnIdentifier )
{
	// After whatever command user has executed - make sure to update toolbar button states
	UpdateOwnToolbarButtonStates();
}
```

In this case `UpdateOwnToolbarButtonStates` is a user function defined in the Global Declarations.

25.3.2.6.1.14 OnUserAddedXMLNode

Event: OnUserAddedXMLNode (XML node)

Description: `OnUserAddedXMLNode` will be fired when the user adds an XML node as a primary action. This happens in the situations, where the user clicks on

- auto-add hyperlinks (see example OnUserAddedXMLNode.sps)
- the Insert…, Insert After…, Insert Before… context menu items
- Append row, Insert row toolbar buttons
- Insert After…, Insert Before… actions in element entry helper (outside StyleVision)

The event doesn't get fired on Duplicate row, or when the node was added externally (e.g. via COM API), or on Apply (e.g. Text State Icons), or when in XML table operations or in DB operations.

The event parameter is the XML node object, which was added giving the user an opportunity to manipulate the XML node added. An elaborate example for an event handler can be found in the `OnUserAddedXMLNode.sps` file.

25.3.2.6.2 Application

Property: Application as Application[1131] (read-only)

Description
Accesses the XMLSpy application object.

Errors

2000	The authentic view object is no longer valid.
2005	Invalid address for the return parameter was specified.

25.3.2.6.3 AsXMLString

Property: AsXMLString as String

Description
Returns or sets the document content as an XML string. Setting the content to a new value does not change the schema file or sps file in use. If the new XMLString does not match the actual schema file error 2011 gets returned.

Errors
2000 The authentic view object is no longer valid.
2011 AsXMLString was set to a value which is no valid XML for the current schema file.

25.3.2.6.4 ContextMenu

Property: ContextMenu() as ContextMenu

Description
The property `ContextMenu` gives access to customize the context menu. The best place to do it is in the event handler `OnContextMenuActivated`.

Errors
2000 Invalid object.
2005 Invalid parameter.

25.3.2.6.5 CreateXMLNode

Method: CreateXMLNode (*nKind* as SPYXMLDataKind 1371) as XMLData 1309

Return Value
The method returns the new XMLData 1309 object.

Description
To create a new XMLData object use the CreateXMLNode() method.

Errors
2000 Invalid object.
2012 Cannot create XML node.

25.3.2.6.6 DisableAttributeEntryHelper

Method: DisableAttributeEntryHelper ()

Description

DisableAttributeEntryHelper () disables the attribute entry helper in XMLSpy, Authentic Desktop and Authentic Browser plug-in.

Errors
2000 Invalid object.

25.3.2.6.7 DisableElementEntryHelper

Method: DisableElementEntryHelper ()

Description
DisableElementEntryHelper () disables the element entry helper in XMLSpy, Authentic Desktop and Authentic Browser plug-in.

Errors
2000 Invalid object.

25.3.2.6.8 DisableEntityEntryHelper

Method: DisableEntityEntryHelper ()

Description
DisableEntityEntryHelper () disables the entity entry helper in XMLSpy, Authentic Desktop and Authentic Browser plug-in.

Errors
2000 Invalid object.

25.3.2.6.9 DocumentBegin

Property: DocumentBegin as AuthenticRange 1157 (read-only)

Description
Retrieve a range object that points to the beginning of the document.

Errors
2000 The authentic view object is no longer valid.
2005 Invalid address for the return parameter was specified.

25.3.2.6.10 DocumentEnd

Property: DocumentEnd as AuthenticRange 1157 (read-only)

(C) 2015-2021 Altova GmbH

Description
Retrieve a range object that points to the end of the document.

Errors
2000 The authentic view object is no longer valid.
2005 Invalid address for the return parameter was specified.

25.3.2.6.11 DoNotPerformStandardAction

Method: DoNotPerformStandardAction ()

Description
`DoNotPerformStandardAction()` serves as cancel bubble for macros, and stops further execution after macro has finished.

Errors
2000 Invalid object.

25.3.2.6.12 EvaluateXPath

Method: EvaluateXPath (XMLData as XMLData 1309, strExpression as string) strValue as string

Return Value
The method returns a string

Description
`EvaluateXPath()` executes an XPath expression with the given XML context node. The result is returned as a string, in the case of a sequence it is a space-separated string.

Errors
2000 Invalid object.
2005 Invalid parameter.
2008 Internal error.
2013 XPath error.

25.3.2.6.13 Event

Property: Event as AuthenticEvent (read-only)

Description
This property gives access to parameters of the last event in the same way as OldAuthenticView.event does. Since all events for the scripting environment and external clients are now available with parameters this Event property should only be used from within IDE-Plugins.

Errors
2000 The authentic view object is no longer valid.
2005 Invalid address for the return parameter was specified.

25.3.2.6.14 EventContext

Property: EventContext() as EventContext

Description
`EventContext` property gives access to the running macros context. See the `EventContext` 1154 interface description for more details.

Errors
2000 Invalid object.

25.3.2.6.15 GetToolbarButtonState

Method: GetToolbarButtonState (`ButtonIdentifier` as string) as `AuthenticToolbarButtonState`

Return Value
The method returns `AuthenticToolbarButtonState`

Description
`Get/SetToolbarButtonState` queries the status of a toolbar button, and lets the user disable or enable the button, identified via its button identifier (see list above 1192). One usage is to disable toolbar buttons permanently. Another usage is to put `SetToolbarButtonState` in the `OnSelectionChanged` event handler, as toolbar buttons are updated regularly when the selection changes in the document.

Toolbar button states are given by the listed enumerations 1360.

The default state means that the enable/disable of the button is governed by AuthenticView. When the user sets the button state to enable or disable, the button remains in that state as long as the user does not change it.

Errors
2000 Invalid object.
2005 Invalid parameter.
2008 Internal error.
2014 Invalid button identifier.

25.3.2.6.16 Goto

Method: Goto (*eKind* as SPYAuthenticElementKind 1360, *nCount* as Long, *eFrom* as *SPYAuthenticDocumentPosition* 1359) as *AuthenticRange* 1157

Description

Retrieve a range object that points to the beginning of the *nCount* element of type *eKind*. The start position is defined by the parameter *eFrom*. Use positive values for *nCount* to navigate to the document end. Use negative values to navigate towards the beginning of the document.

Errors

2000	The authentic view object is no longer valid.
2003	Target lies after end of document.
2004	Target lies before beginning of document.
2005	Invalid element kind specified. The document position to start from is not one of *spyAuthenticDocumentBegin* or *spyAuthenticDocumentEnd*. Invalid address for the return parameter was specified.

Examples

```
' ----------------------------------------
' Scripting environment - VBScript
' ----------------------------------------
Dim objAuthenticView
' we assume that the active document is open in authentic view mode
Set objAuthenticView = Application.ActiveDocument.AuthenticView

On Error Resume Next
Dim objRange
' goto beginning of first table in document
Set objRange = objAuthenticView.Goto (spyAuthenticTable, 1, spyAuthenticDocumentBegin)
If (Err.number = 0) Then
	objRange.Select()
Else
	MsgBox "No table found in document"
End If
```

25.3.2.6.17 IsRedoEnabled

Property: IsRedoEnabled as Boolean (read-only)

Description

True if redo steps are available and Redo 1201 is possible.

Errors

2000	The authentic view object is no longer valid.
2005	Invalid address for the return parameter was specified.

25.3.2.6.18 IsUndoEnabled

Property: IsUndoEnabled as Boolean (read-only)

Description

True if undo steps are available and Undo 1202 is possible.

Errors

2000 The authentic view object is no longer valid.
2005 Invalid address for the return parameter was specified.

25.3.2.6.19 MarkupVisibility

Property: MarkupVisibility as SPYAuthenticMarkupVisibility 1360

Description
Set or get current visibility of markup.

Errors

2000 The authentic view object is no longer valid.
2005 Invalid enumeration value was specified.
Invalid address for the return parameter was specified.

25.3.2.6.20 Parent

Property: Parent as Document 1218 (read-only)

Description
Access the document shown in this view.

Errors

2000 The authentic view object is no longer valid.
2005 Invalid address for the return parameter was specified.

25.3.2.6.21 Print

Method: Print (*bWithPreview* as Boolean, *bPromptUser* as Boolean)

Description
Print the document shown in this view. If *bWithPreview* is set to *True*, the print preview dialog pops up. If *bPromptUser* is set to *True*, the print dialog pops up. If both parameters are set to *False*, the document gets printed without further user interaction.

Errors

2000 The authentic view object is no longer valid.

(C) 2015-2021 Altova GmbH

25.3.2.6.22 Redo

Method: Redo() as Boolean

Description
Redo the modification undone by the last undo command.

Errors
2000 The authentic view object is no longer valid.
2005 Invalid address for the return parameter was specified.

25.3.2.6.23 Selection

Property: Selection as AuthenticRange 1157

Description
Set or get current text selection in user interface.

Errors
2000 The authentic view object is no longer valid.
2002 No cursor selection is active.
2005 Invalid address for the return parameter was specified.

Examples

```
' ---------------------------------------
' Scripting environment - VBScript
' ---------------------------------------
Dim objAuthenticView
' we assume that the active document is open in authentic view mode
Set objAuthenticView = Application.ActiveDocument.AuthenticView

' if we are the end of the document, re-start at the beginning
If (objAuthenticView.Selection.IsEqual(objAuthenticView.DocumentEnd)) Then
        objAuthenticView.Selection = objAuthenticView.DocumentBegin
Else
        ' objAuthenticView.Selection = objAuthenticView.Selection.GotoNextCursorPosition()
        ' or shorter:
         objAuthenticView.Selection.GotoNextCursorPosition().Select
End If
```

25.3.2.6.24 SetToolbarButtonState

Method: SetToolbarButtonState (`ButtonIdentifier` as string, `AuthenticToolbarButtonState` state)

Description
`Get/SetToolbarButtonState` queries the status of a toolbar button, and lets the user disable or enable the button, identified via its button identifier (see list above 1192). One usage is to disable toolbar buttons permanently. Another usage is to put `SetToolbarButtonState` in the `OnSelectionChanged` event handler, as toolbar buttons are updated regularly when the selection changes in the document.

Toolbar button states are given by the listed enumerations 1360.

The default state means that the enable/disable of the button is governed by AuthenticView. When the user sets the button state to enable or disable, the button remains in that state as long as the user does not change it.

Errors

2000 Invalid object.
2008 Internal error.
2014 Invalid button identifier.

25.3.2.6.25 Undo

Method: Undo() as Boolean

Description
Undo the last modification of the document from within this view.

Errors

2000 The authentic view object is no longer valid.
2005 Invalid address for the return parameter was specified.

25.3.2.6.26 UpdateXMLInstanceEntities

Method: UpdateXMLInstanceEntities()

Description
Updates the internal representation of the declared entities, and refills the entry helper. In addition, the validator is reloaded, allowing the XML file to validate correctly. Please note that this may also cause schema files to be reloaded.

Errors
The method never returns an error.

Example

```
// ----------------------------------------
// Scripting environment - JavaScript
// ----------------------------------------
if(Application.ActiveDocument && (Application.ActiveDocument.CurrentViewMode == 4))
{
	var objDocType;
	objDocType = Application.ActiveDocument.DocEditView.XMLRoot.GetFirstChild(10);

	if(objDocType)
	{
		var objEntity = Application.ActiveDocument.CreateChild(14);
		objEntity.Name = "child";
		objEntity.TextValue = "SYSTEM \"child.xml\"";
```

```
                objDocType.AppendChild(objEntity);

                Application.ActiveDocument.AuthenticView.UpdateXMLInstanceEntities();
        }
}
```

25.3.2.6.27 WholeDocument

Property: WholeDocument as AuthenticRange 1157 (read-only)

Description
Retrieve a range object that selects the whole document.

Errors
2000 The authentic view object is no longer valid.
2005 Invalid address for the return parameter was specified.

25.3.2.6.28 XMLDataRoot

Property: XMLDataRoot as XMLData 1309 (read-only)

Description
Returns or sets the top-level XMLData element of the current document. This element typically describes the document structure and would be of kind spyXMLDataXMLDocStruct, spyXMLDataXMLEntityDocStruct or spyXMLDataDTDDocStruct..

Errors
2000 The authentic view object is no longer valid.
2005 Invalid address for the return parameter was specified.

25.3.2.7 CodeGeneratorDlg

Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

Properties and Methods

Standard automation properties
Application 1204
Parent 1208

Programming language selection properties
ProgrammingLanguage 1208
TemplateFileName 1209

Settings for C++ code
CPPSettings_DOMType 1205
CPPSettings_LibraryType 1206
CPPSettings_UseMFC 1206

CPPSettings_GenerateVC6ProjectFile 1205
CPPSettings_GenerateVSProjectFile 1206

Settings for C# code
CSharpSettings_ProjectType 1206

Dialog handling for above code generation properties
PropertySheetDialogAction 1208

Output path selection properties
OutputPath 1207
OutputPathDialogAction 1207

Presentation of result
OutputResultDialogAction 1207

Description
Use this object to configure the generation of program code for schema files. The method GenerateProgramCode 1234 expects a CodeGeneratorDlg as parameter to configure code generation as well as the associated user interactions.

25.3.2.7.1 Application

Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

Property: Application as Application 1131 (read-only)

Description
Access the XMLSpy application object.

Errors
2200 The object is no longer valid.
2201 Invalid address for the return parameter was specified.

25.3.2.7.2 CompatibilityMode (obsolete)

Property: CompatibilityMode as Boolean
Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

Description
Set to true to generate code compatible to XMLSpy 2005R3. Set to false to use newly added code-generation features.

Errors
2200 The object is no longer valid.
2201 Invalid action passed as parameter or an invalid address was specified for the return parameter.

(C) 2015-2021 Altova GmbH

25.3.2.7.3 CPPSettings_DOMType

Property: CPPSettings_DOMType as SPYDOMType 1361
Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

Description
Defines one of the settings that configure generation of C++ code.

Errors
2200 The object is no longer valid.
2201 Invalid action passed as parameter or an invalid address was specified for the return parameter.

25.3.2.7.4 CPPSettings_GenerateVC6ProjectFile

Property: CPPSettings_GenerateVC6ProjectFile as Boolean
Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

Description
Defines one of the settings that configure generation of C++ code.

Errors
2200 The object is no longer valid.
2201 Invalid action passed as parameter or an invalid address was specified for the return parameter.

25.3.2.7.5 CPPSettings_GenerateGCCMakefile

Property: CPPSettings_GenerateGCCMakefile as Boolean
Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

Description
Creates makefiles to compile the generated code under Linux with GCC.

Errors
2200 The object is no longer valid.
2201 Invalid action passed as parameter or an invalid address was specified for the return parameter.

25.3.2.7.6 CPPSettings_GenerateVSProjectFile

Property: CSharpSettings_GenerateVSProjectFile as SPYProjectType 1366
Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

Description
Defines one of the settings that configure generation of C++ code. Only `spyVisualStudio2005Project (=4)` and `spyVisualStudio2008Project (=5)` and `spyVisualStudio2010Project (=6)` are valid project types.

Errors
2200 The object is no longer valid.
2201 Invalid action passed as parameter or an invalid address was specified for the return parameter.

25.3.2.7.7 CPPSettings_LibraryType

Property: CPPSettings_LibraryType as SPYLibType 1364
Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

Description
Defines one of the settings that configure generation of C++ code.

Errors
2200 The object is no longer valid.
2201 Invalid action passed as parameter or an invalid address was specified for the return parameter.

25.3.2.7.8 CPPSettings_UseMFC

Property: CPPSettings_UseMFC as Boolean
Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

Description
Defines one of the settings that configure generation of C++ code.

Errors
2200 The object is no longer valid.
2201 Invalid action passed as parameter or an invalid address was specified for the return parameter.

25.3.2.7.9 CSharpSettings_ProjectType

Property: CSharpSettings_ProjectType as SPYProjectType 1366
Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

(C) 2015-2021 Altova GmbH

Description
Defines the only setting to configure generation of C# code.

Errors
2200 The object is no longer valid.
2201 Invalid action passed as parameter or an invalid address was specified for the return parameter.

25.3.2.7.10 OutputPath

Property: OutputPath as String
Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

Description
Selects the base directory for all generated code.

Errors
2200 The object is no longer valid.
2201 Invalid address for the return parameter was specified.

25.3.2.7.11 OutputPathDialogAction

Property: OutputPathDialogAction as SPYDialogAction 1361
Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

Description
Defines how the sub-dialog for selecting the code generation output path gets handled. Set this value to *spyDialogUserInput(2)* to show the dialog with the current value of the OutputPath 1207 property as default. Use *spyDialogOK(0)* to hide the dialog from the user.

Errors
2200 The object is no longer valid.
2201 Invalid action passed as parameter or an invalid address was specified for the return parameter.

25.3.2.7.12 OutputResultDialogAction

Property: OutputResultDialogAction as SPYDialogAction 1361
Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

Description
Defines how the sub-dialog that asks to show the result of the code generation process gets handled. Set this value to *spyDialogUserInput(2)* to show the dialog. Use *spyDialogOK(0)* to hide the dialog from the user.

Errors

2200 The object is no longer valid.
2201 Invalid action passed as parameter or an invalid address was specified for the return parameter.

25.3.2.7.13 Parent

Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

Property: Parent as Dialogs 1215 (read-only)

Description
Access the parent of the object.

Errors
2200 The object is no longer valid.
2201 Invalid address for the return parameter was specified.

25.3.2.7.14 ProgrammingLanguage

Property: ProgrammingLanguage as ProgrammingLanguage 1365
Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

Description
Selects the output language for the code to be generated.

CAUTION: Setting this property to one of C++, C# or Java, changes the property TemplateFileName 1209 to the appropriate template file delivered with XMLSpy as well. If you want to generate C++, C# or Java code based on your own templates, set first the programming language and then select your template file.

Errors
2200 The object is no longer valid.
2201 Invalid address for the return parameter was specified.

25.3.2.7.15 PropertySheetDialogAction

Property: PropertySheetDialogAction as SPYDialogAction 1361
Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

Description
Defines how the sub-dialog that configures the code generation process gets handled. Set this value to *spyDialogUserInput(2)* to show the dialog with the current values as defaults. Use *spyDialogOK(0)* to hide the dialog from the user.

Errors
2200 The object is no longer valid.

2201 Invalid action passed as parameter or an invalid address was specified for the return parameter.

25.3.2.7.16 TemplateFileName

Property: TemplateFileName as String
Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

Description
Selects the code generation template file. XMLSpy comes with template files for C++, C# or Java in the SPL folder of your installation directory.

Setting this property to one of the code generation template files of your XMLSpy installation automatically sets the ProgrammingLanguage 1208 property to its appropriate value.

Errors
2200 The object is no longer valid.
2201 Invalid address for the return parameter was specified.

25.3.2.8 DatabaseConnection

Properties for import and export
File 1212 or
ADOConnection 1210 or
ODBCConnection 1213

Properties for import only
DatabaseKind 1211
SQLSelect 1214
AsAttributes 1210
ExcludeKeys 1212
IncludeEmptyElements 1213
NumberDateTimeFormat 1213
NullReplacement 1213
CommentIncluded 1210

Properties for export only
CreateMissingTables 1211
CreateNew 1211
TextFieldLen 1215
DatabaseSchema 1211

Properties for XML Schema from DB Structure generation
PrimaryKeys 1214
ForeignKeys 1212
UniqueKeys 1215
SchemaExtensionType 1214
SchemaFormat 1214
ImportColumnsType 1212

Description
DatabaseConnection specifies the parameters for the database connection.

Please note that the properties of the DatabaseConnection interface are referring to the settings of the import and export dialogs of XMLSpy.

25.3.2.8.1 ADOConnection

Property: ADOConnection as String

Description
The property ADOConnection contains a connection string. Either use this property or ODBCConnection[1213] or File[1212] to refer to a database.

Errors
No error codes are returned.

Example

```
Dim objSpyConn As DatabaseConnection
Set objSpyConn = objSpy.GetDatabaseSettings

Dim objADO As DataLinks
Set objADO = CreateObject("DataLinks")

If Not (objADO Is Nothing) Then
    Dim objConn As Connection
    Set objConn = objADO.PromptNew
    objSpyConn.ADOConnection = objConn.ConnectionString
End If
```

25.3.2.8.2 AsAttributes

Property: AsAttributes as Boolean

Description
Set AsAttributes to true if you want to initialize all import fields to be imported as attributes. Default is false and will initialize all fields to be imported as elements. This property is used only in calls to Application.GetDatabaseImportElementList[1137].

Errors
No error codes are returned.

25.3.2.8.3 CommentIncluded

Property: CommentIncluded as Boolean

Description

This property tells whether additional comments are added to the generated XML. Default is true. This property is used only when importing from databases.

Errors
No error codes are returned.

25.3.2.8.4 CreateMissingTables

Property: CreateMissingTables as Boolean

Description
If CreateMissingTables is true, tables which are not already defined in the export database will be created during export. Default is true. This property is used only when exporting to databases.

Errors
No error codes are returned.

25.3.2.8.5 CreateNew

Property: CreateNew as Boolean

Description
Set CreateNew true if you want to create a new database on export. Any existing database will be overwritten. See also DatabaseConnection.File 1212. Default is false. This property is used only when exporting to databases.

Errors
No error codes are returned.

25.3.2.8.6 DatabaseKind

Property: DatabaseKind as SPYDatabaseKind 1360

Description
Select the kind of database that gets access. The default value is spyDB_Unspecified(7) and is sufficient in most cases. This property is used only when importing from databases.

Errors
No error codes are returned.

25.3.2.8.7 DatabaseSchema

Property: DatabaseSchema as String

Description
This property specifies the Schema used for export in Schema aware databases. Default is "". This property is used only when exporting to databases.

Errors
No error codes are returned.

25.3.2.8.8 ExcludeKeys

Property: ExcludeKeys as Boolean

Description
Set ExcludeKeys to true if you want to exclude all key columns from the import data. Default is false. This property is used only when importing from databases.

Errors
No error codes are returned.

25.3.2.8.9 File

Property: File as String

Description
The property File sets the path for the database during export or import. This property can only be used in conjunction with a Microsoft Access database. Either use this property or ODBCConnection[1213] or ADOConnection[1210] to refer to the database.

See also Import and Export[1103].

Errors
No error codes are returned.

25.3.2.8.10 ForeignKeys

Property: ForeignKeys as Boolean

Description
Specifies whether the Foreign Keys constraint is created or not. Default is true. This property is used only when creating a XML Schema from a DB structure.

Errors
No error codes are returned.

25.3.2.8.11 ImportColumnsType

Property: ImportColumnsType as SPYImportColumnsType[1363]

Description
Defines if column information from the DB is saved as element or attribute in the XML Schema. Default is as element. This property is used only when creating a XML Schema from a DB structure.

Errors
No error codes are returned.

(C) 2015-2021 Altova GmbH

25.3.2.8.12 IncludeEmptyElements

Property: IncludeEmptyElements as Boolean

Description
Set IncludeEmptyElements to false if you want to exclude all empty elements. Default is true. This property is used only when importing from databases.

Errors
No error codes are returned.

25.3.2.8.13 NullReplacement

Property: NullReplacement as String

Description
This property contains the text value that is used during import for empty elements (null values). Default is "". This property is used only when importing from databases.

Errors
No error codes are returned.

25.3.2.8.14 NumberDateTimeFormat

Property: NumberDateTimeFormat as SPYNumberDateTimeFormat 1365

Description
The property NumberDateTimeFormat sets the format of numbers and date- and time-values. Default is spySystemLocale 1365. This property is used only when importing from databases.

Errors
No error codes are returned.

25.3.2.8.15 ODBCConnection

Property: ODBCConnection as String

Description
The property ODBCConnection contains a ODBC connection string. Either use this property or ADOConnection 1210 or File 1212 to refer to a database.

Errors
No error codes are returned.

25.3.2.8.16 PrimaryKeys

Property: PrimaryKeys as Boolean

Description
Specifies whether the Primary Keys constraint is created or not. Default is true. This property is used only when creating a XML Schema from a DB structure.

Errors
No error codes are returned.

25.3.2.8.17 SchemaExtensionType

Property: SchemaExtensionType as SPYSchemaExtensionType 1368

Description
Defines the Schema extension type used during the Schema generation. This property is used only when creating a XML Schema from a DB structure.

See also Create XML Schema from DB Structure 946.

Errors
No error codes are returned.

25.3.2.8.18 SchemaFormat

Property: SchemaFormat as SPYSchemaFormat 1368

Description
Defines the Schema format used during the Schema generation. This property is used only when creating a XML Schema from a DB structure.

See also Create XML Schema from DB Structure 946.

Errors
No error codes are returned.

25.3.2.8.19 SQLSelect

Property: SQLSelect as String

Description
The SQL query for the import is stored in the property SQLSelect. This property is used only when importing from databases. See also Import and Export 1103.

Errors
No error codes are returned.

25.3.2.8.20 TextFieldLen

Property: TextFieldLen as long

Description
The property TextFieldLen sets the length for created text fields during the export. Default is 255. This property is used only when exporting to databases.

Errors
No error codes are returned.

25.3.2.8.21 UniqueKeys

Property: UniqueKeys as Boolean

Description
Specifies whether the Unique Keys constraint is created or not. Default is true. This property is used only when creating a XML Schema from a DB structure.

Errors
No error codes are returned.

25.3.2.9 Dialogs

Properties and Methods

Standard automation properties
Application 1215
Parent 1216

Various dialog objects
CodeGeneratorDlg 1216
FileSelectionDlg 1216
SchemaDocumentationDlg 1217
GenerateSampleXMLDlg 1217
DTDSchemaGeneratorDlg 1217
FindInFilesDlg 1217

Description
The Dialogs object provides access to different built-in dialogs of XMLSpy. These dialog objects allow to initialize the fields of user dialogs before they get presented to the user or allow to simulate complete user input by your program.

25.3.2.9.1 Application

Property: Application as Application 1131 (read-only)

Description

Access the XMLSpy application object.

Errors
2300 The object is no longer valid.
2301 Invalid address for the return parameter was specified.

25.3.2.9.2 CodeGeneratorDlg

Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

Property: CodeGeneratorDlg as CodeGeneratorDlg 1203 (read-only)

Description
Get a new instance of a code generation dialog object. You will need this object to pass the necessary parameters to the code generation methods. Initial values are taken from last usage of the code generation dialog.

Errors
2300 The Dialogs object or one of its parents is no longer valid.
2301 Invalid address for the return parameter was specified.

25.3.2.9.3 FileSelectionDlg

Property: FileSelectionDlg as FileSelectionDlg 1260 (read-only)

Description
Get a new instance of a file selection dialog object.

File selection dialog objects are passed to you with the some events that signal opening or saving of documents and projects.

Errors
2300 The Dialogs object or one of its parents is no longer valid.
2301 Invalid address for the return parameter was specified.

25.3.2.9.4 Parent

Property: Parent as Application 1131 (read-only)

Description
Access the XMLSpy application object.

Errors
2300 The object is no longer valid.
2301 Invalid address for the return parameter was specified.

(C) 2015-2021 Altova GmbH

25.3.2.9.5 SchemaDocumentationDlg

Property: SchemaDocumentationDlg as SchemaDocumentationDlg 1282 (read-only)

Description
Get a new instance of a dialog object that parameterizes generation of schema documentation. See Document.GenerateSchemaDocumentation 1234 for its usage.

Errors
2300 The Dialogs object or one of its parents is no longer valid.
2301 Invalid address for the return parameter was specified.

25.3.2.9.6 GenerateSampleXMLDlg

Property: GenerateSampleXMLDlg as GenerateSampleXMLDlg 1273 (read-only)

Description
Get a new instance of a dialog object that parameterizes generation of a sample XML based on a W3C schema or DTD. See GenerateSampleXML 1234 for its usage.

Errors
2300 The Dialogs object or one of its parents is no longer valid.
2301 Invalid address for the return parameter was specified.

25.3.2.9.7 DTDSchemaGeneratorDlg

Property: DTDSchemaGeneratorDlg as DTDSchemaGeneratorDlg 1251 (read-only)

Description
Get a new instance of a dialog object that parameterizes generation of a schema or DTD. See Document.GenerateDTDOrSchemaEx 1233 for its usage.

Errors
2300 The Dialogs object or one of its parents is no longer valid.
2301 Invalid address for the return parameter was specified.

25.3.2.9.8 FindInFilesDlg

Property: FindInFilesDlg as FindInFilesDlg 1261 (read-only)

Description
Get a new instance of a dialog object that parameterizes the search (or replacement) of strings in files. See Application.FindInFiles 1137 for its usage.

Errors
2300 The Dialogs object or one of its parents is no longer valid.

2301 Invalid address for the return parameter was specified.

25.3.2.10 Document

Properties and Methods

Standard automation properties
Application 1222
Parent 1239

Various document properties and methods
SetActiveDocument 1242
Encoding 1229
SetEncoding (obsolete) 1242
Suggestions 1244

XML validation
IsValid 1236
SetExternalIsValid 1243

Document conversion and transformation
AssignDTD 1222
AssignSchema 1223
AssignXSL 1223
AssignXSLFO 1223
ConvertDTDOrSchema 1225
ConvertDTDOrSchemaEx 1225
GenerateDTDOrSchema 1233
GenerateDTDOrSchemaEx 1233
FlattenDTDOrSchema 1232
CreateSchemaDiagram 1227
ExecuteXQuery 1229
TransformXSL 1245
TransformXSLEx 1245
TransformXSLFO 1245
GenerateProgramCode 1234 (Enterprise Edition only)
GenerateSchemaDocumentation 1234
GenerateSampleXML 1234
ConvertToWSDL20 1226

Document export
GetExportElementList 1235
ExportToText 1231
ExportToDatabase 1230
CreateDBStructureFromXMLSchema 1227
GetDBStructureList 1235

File saving and naming
FullName 1232
Name 1239
Path 1240
GetPathName (obsolete) 1236
SetPathName (obsolete) 1243

(C) 2015-2021 Altova GmbH

Title 1245
IsModified 1236
Saved 1241
SaveAs 1241
Save 1240
SaveInString 1241
SaveToURL 1242
Close 1225

View access
CurrentViewMode 1228
SwitchViewMode 1244
AuthenticView 1224
GridView 1236
DocEditView (obsolete) 1228

Access to XMLData
RootElement 1240
DataRoot 1228
CreateChild 1227
UpdateViews 1246
StartChanges 1244
EndChanges 1229
UpdateXMLData 1246

Description
Document objects represent XML documents opened in XMLSpy.

Use one of the following properties to access documents that are already open XMLSpy:
Application.ActiveDocument 1134
Application.Documents 1136

Use one of the following methods to open a new document in XMLSpy:
Documents.OpenFile 1249
Documents.OpenURL 1250
Documents.OpenURLDialog 1250
Documents.NewFile 1248
Documents.NewFileFromText 1249
SpyProjectItem.Open 1298
Application.ImportFromDatabase 1141
Application.ImportFromSchema 1142
Application.ImportFromText 1143
Application.ImportFromWord 1144
Document.ConvertDTDOrSchema 1225
Document.GenerateDTDOrSchema 1233

25.3.2.10.1 Events

25.3.2.10.1.1 OnBeforeSaveDocument

Event: OnBeforeSaveDocument(*objDocument* as Document[1218], *objDialog* as FileSelectionDlg[1260])

XMLSpy scripting environment - VBScript:
Function On_BeforeSaveDocument(*objDocument*, *objDialog*)
End Function

' old handler - now obsolete
' return string to save to new file name
' return empty string to cancel save operation
' return nothing to save to original name
Function On_SaveDocument(*objDocument*, *strFilePath*)
End Function

XMLSpy scripting environment - JScript:
function On_BeforeSaveDocument(*objDocument*, *objDialog*)
{
}

// old handler - now obsolete
// return string to save to new file name
// return empty string to cancel save operation
// return nothing to save to original name
function On_SaveDocument(*objDocument*, *strFilePath*)
{
}

XMLSpy IDE Plugin:
IXMLSpyPlugIn.OnEvent (27, ...)// nEventId = 27

Description
This event gets fired on any attempt to save a document. The file selection dialog object is initialized with the name chosen for the document file. You can modify this selection. To continue saving the document leave the FileSelectionDlg.DialogAction[1260] property of *io_objDialog* at its default value spyDialogOK[1361]. To abort saving of the document set this property to spyDialogCancel[1361].

25.3.2.10.1.2 OnBeforeCloseDocument

Event: OnBeforeCloseDocument(*objDocument* as Document[1218])as Boolean

XMLSpy scripting environment - VBScript:
Function On_BeforeCloseDocument(*objDocument*)
 ' On_BeforeCloseDocument = False ' to prohibit closing of document
End Function

XMLSpy scripting environment - JScript:
function On_BeforeCloseDocument(*objDocument*)

```
{
        // return false;  /* to prohibit closing of document */
}
```

XMLSpy IDE Plugin:
IXMLSpyPlugIn.OnEvent (28, ...)// nEventId = 28

Description
This event gets fired on any attempt to close a document. To prevent the document from being closed return false.

25.3.2.10.1.3 OnBeforeValidate

Event: OnBeforeValidate(*objDocument* as Document[1218], *bOnLoading* as Boolean, *bOnCommand* as Boolean) as Boolean

XMLSpy scripting environment - VBScript:

```
Function On_BeforeValidate(objDocument, bOnLoading, bOnCommand)
        On_BeforeValidate = bCancelDefaultValidation    'set by the script if necessary
End Function
```

XMLSpy scripting environment - JScript:

```
function On_BeforeValidate(objDocument, bOnLoading, bOnCommand)
{
        return bCancelDefaultValidation  //set by the script if necessary
}
```

XMLSpy IDE Plugin:
IXMLSpyPlugIn.OnEvent (32, ...)// nEventId = 32

Description
This event gets fired before the document is validated. It is possible to suppress the default validation by returning false from the event handler. In this case the script should also set the validation result using the SetExternalIsValid[1243] method.

bOnLoading is true if the event is raised on the initial validation on loading the document.

bOnCommand is true whenever the user selected the Validate command from the Toolbar or menu.

Available with TypeLibrary version 1.5

25.3.2.10.1.4 OnCloseDocument

Event: OnCloseDocument(*objDocument* as Document[1218])

XMLSpy scripting environment - VBScript:

```
Function On_Close Document(objDocument)
End Function
```

XMLSpy scripting environment - JScript:

function On_Close Document(*objDocument*)
{
}

XMLSpy IDE Plugin:
IXMLSpyPlugIn.OnEvent (8, ...) // nEventId = 8

Description
This event gets fired as a result of closing a document. Do not modify the document from within this event.

25.3.2.10.1.5 OnViewActivation

Event: OnViewActivation(*objDocument* as Document[1218], *eViewMode* as SPYViewModes[1370], *bActivated* as Boolean)

XMLSpy scripting environment - VBScript:
Function On_ViewActivation(*objDocument*, *eViewMode*, *bActivated*)
End Function

XMLSpy scripting environment - JScript:
function On_ViewActivation(*objDocument*, *eViewMode*, *bActivated*)
{
}

XMLSpy IDE Plugin:
IXMLSpyPlugIn.OnEvent (29, ...)// nEventId = 29

Description
This event gets fired whenever a view of a document becomes visible (i.e. becomes the active view) or invisible (i.e. another view becomes the active view or the document gets closed). However, the first view activation event after a document gets opened cannot be received, since there is no document object to get the event from. Use the Application.OnDocumentOpened[1133] event instead.

25.3.2.10.2 Application

Property: Application as Application[1131] (read-only)

Description
Accesses the XMLSpy application object.

Errors
1400 The object is no longer valid.
1407 Invalid address for the return parameter was specified.

25.3.2.10.3 AssignDTD

Method: AssignDTD(*strDTDFile* as String, *bDialog* as Boolean)

(C) 2015-2021 Altova GmbH

Description
The method places a reference to the DTD file "strDTDFile" into the document. Note that no error occurs if the file does not exist, or is not accessible. If bDialog is true XMLSpy presents a dialog to set the file.

Errors
1400 The object is no longer valid.
1409 You are not allowed to assign a DTD to the document.

25.3.2.10.4 AssignSchema

Method: AssignSchema (*strSchemaFile* as String, *bDialog* as Boolean)

Description
The method places a reference to the schema file "strSchemaFile" into the document. Note that no error occurs if the file does not exist or is not accessible. If bDialog is true XMLSpy presents a dialog to set the file.

Errors
1400 The object is no longer valid.
1409 You are not allowed to assign a schema file to the document.

25.3.2.10.5 AssignXSL

Method: AssignXSL (*strXSLFile* as String, *bDialog* as Boolean)

Description
The method places a reference to the XSL file "strXSLFile" into the document. Note that no error occurs if the file does not exist or is not accessible. If bDialog is true XMLSpy presents a dialog to set the file.

Errors
1400 The object is no longer valid.
1409 You are not allowed to assign an XSL file to the document.

25.3.2.10.6 AssignXSLFO

Method: AssignXSLFO (*strXSLFOFile* as String, *bDialog* as Boolean)

Description
The method places a reference to the XSLFO file "strXSLFile" into the document. Note that no error occurs if the file does not exist or is not accessible. If bDialog is true XMLSpy presents a dialog to set the file.

Errors
1400 The object is no longer valid.
1409 You are not allowed to assign an XSL file to the document.

25.3.2.10.7 AsXMLString

Property: AsXMLString as String

Description
This property can be used to get or set the document content.

Errors
1400 The document object is no longer valid.
1404 Cannot create XMLData object.
1407 View mode cannot be switched.

25.3.2.10.8 AuthenticView

Method: AuthenticView as AuthenticView[1185] (read-only)

Description
Returns an object that gives access to properties and methods specific to Authentic view. The object returned is only valid if the current document is opened in Authentic view mode. The lifetime of an object ends with the next view switch. Any attempt to access objects or any of its children afterwards will result in an error indicating that the object is invalid.

AuthenticView[1185] and DocEditView[1335] both provide automation access to the Authentic view mode of XMLSpy. Functional overlap is intentional. A future version of Authentic View will include all functionality of DocEditView[1335] and its sub-objects, thereby making usage of DocEditView[1335] obsolete.

Errors
1400 The object is no longer valid.
1417 Document needs to be open in authentic view mode.

Examples

```
' ----------------------------------------
' XMLSpy scripting environment - VBScript
' secure access to authentic view object
' ----------------------------------------
Dim objDocument
Set objDocument = Application.ActiveDocument
If (Not objDocument Is Nothing) Then
	' we have an active document, now check for view mode
	If (objDocument.CurrentViewMode <> spyViewAuthentic) Then
		If (Not objDocument.SwitchViewMode (spyViewAuthentic)) Then
			MsgBox "Active document does not support authentic view mode"
		Else
			' now it is safe to access the authentic view object
			Dim objAuthenticView
			Set objAuthenticView = objDocument.AuthenticView
			' now use the authentic view object

		End If
	End If
```

```
Else
        MsgBox "No document is open"
End If
```

25.3.2.10.9 Close

Method: Close (*bDiscardChanges* as Boolean)

Description
To close the document call this method. If bDiscardChanges is true and the document is modified, the document will be closed but not saved.

Errors
1400 The object is no longer valid.
1401 Document needs to be saved first.

25.3.2.10.10 ConvertDTDOrSchema

Method: ConvertDTDOrSchema (*nFormat* as SPYDTDSchemaFormat[1361], *nFrequentElements* as SPYFrequentElements[1362])

Parameters
nFormat
Sets the schema output format to DTD or W3C.

nFrequentElements
Create complex elements as elements or complex types.

Description
ConvertDTDOrSchema takes an existing schema format and converts it into a different format.
For a finer tuning of DTD/XSD conversion, use ConvertDTDOrSchemaEx[1225].

Errors
1400 The object is no longer valid.
1412 Error during conversion. In the case of DTD to DTD or XSD to XSD conversion, the following errors are returned: *DTD to DTD conversion is not supported. Please use function FlattenDTDOrSchema instead* and *Schema to schema conversion is not supported. Please use function FlattenDTDOrSchema instead*.

25.3.2.10.11 ConvertDTDOrSchemaEx

Method: ConvertDTDOrSchemaEx (*nFormat* as SPYDTDSchemaFormat[1361], *nFrequentElements* as SPYFrequentElements[1362], sOutputPath as String, nOutputPathDialogAction as SPYDialogAction[1361])

Parameters
nFormat
Sets the schema output format to DTD, or W3C.

nFrequentElements
Create complex elements as elements or complex types.

`sOutputPath`
The file path for the newly generated file.

`nOutputPathDialogAction`
Defines the dialog interaction for this call.

Description
`ConvertDTDOrSchemaEx` takes an existing schema format and converts it into a different format.

Errors

1400	The object is no longer valid.
1412	Error during conversion. In the case of DTD to DTD or XSD to XSD conversion, the following errors are returned: *DTD to DTD conversion is not supported. Please use function FlattenDTDOrSchema instead* and *Schema to schema conversion is not supported. Please use function FlattenDTDOrSchema instead*.

25.3.2.10.12 ConvertToWSDL20

Method: ConvertToWSDL20 (*sFilePath* as String, *bShowDialogs* as Boolean)

Parameters
sFilePath
This specifies the file name of the converted WSDL. In case the source WSDL includes files which also must be converted, then only the directory part of the given path is used and the file names are generated automatically.

bShowDialogs
Defines whether file/folder selection dialogs are shown.

Description
Converts the WSDL 1.1 document to a WSDL 2.0 file. It will also convert any referenced WSDL files that are referenced from within this document. Note that this functionality is limited to WSDL View only. See Document.CurrentViewMode 1228. and SPYViewModes 1370.

Errors

1400	The document object is no longer valid.
1407	Invalid parameters have been passed or an empty file name has been specified as output target.
1417	The document is not opened in WSDL view, maybe it is not an '.wsdl' file.
1421	Feature is not available in this edition.

(C) 2015-2021 Altova GmbH

1433 WSDL 1.1 to WSDL 2.0 conversion failed.

25.3.2.10.13 CreateChild

Method: CreateChild (*nKind* as SPYXMLDataKind[1371]) as XMLData[1309]

Return Value
The method returns the new XMLData object.

Description
To create a new XMLData object use the CreateChild() method.

Errors
1400 The object is no longer valid.
1404 Cannot create XMLData object.
1407 Invalid address for the return parameter was specified.

25.3.2.10.14 CreateDBStructureFromXMLSchema

Method: CreateDBStructureFromXMLSchema (*pDatabase* as DatabaseConnection[1209], *pTables* as ElementList[1255], *bDropTableWithExistingName* as Boolean) as String

Description
CreateDBStructureFromXMLSchema exports the given tables to the specified database. The function returns the SQL statements that were necessary to perform the changes.

See also GetDBStructureList[1235].

Errors
1429 Database selection missing.
1430 Document export failed.

25.3.2.10.15 CreateSchemaDiagram

Method: CreateSchemaDiagram (*nKind* as SPYSchemaDefKind[1367], *strName* as String, *strFile* as String)

Return Value
None.

Description

The method creates a diagram of the schema type strName of kind nKind and saves the output file into strFile. Note that this functionality is limited to Schema View only. See Document.CurrentViewMode[1228]. and SPYViewModes[1370].

Errors

1400 The object is no longer valid.
1414 Failed to save diagram.
1415 Invalid schema definition type specified.

25.3.2.10.16 CurrentViewMode

Method: CurrentViewMode as SPYViewModes[1370]

Description

The property holds the current view mode of the document. See also Document.SwitchViewMode[1244].

Errors

1400 The object is no longer valid.
1407 Invalid address for the return parameter was specified.

25.3.2.10.17 DataRoot

Property: DataRoot as XMLData[1309] (read-only)

Description

This property provides access to the document's first XMLData object of type *spyXMLDataElement*. This is typically the root element for all document content data. See XMLSpyDocument.RootElement[1240] to get the root element of the whole document including XML prolog data. If the CurrentViewMode[1228] is not *spyViewGrid* or *spyViewAuthentic* an UpdateXMLData[1246] may be necessary to get access to the latest XMLData[1309].

Errors

1400 The document object is no longer valid.
1407 Invalid address for the return parameter was specified.

25.3.2.10.18 DocEditView

Method: DocEditView as DocEditView

Description

Holds a reference to the current Authentic View object.

Errors

1400 The object is no longer valid.
1407 Invalid address for the return parameter was specified.
1417 Document needs to be open in authentic view mode.

(C) 2015-2021 Altova GmbH

25.3.2.10.19 Encoding

Property: Encoding as String

Description
This property provides access to the document's encoding value. However, this property can only be accessed when the document is opened in *spyViewGrid*, *spyViewText* or *spyViewAuthentic*. See CurrentViewMode 1228 on how to detect a document's actual view mode.

This property makes the method SetEncoding 1242 obsolete.

Possible values are, for example:

8859-1,
8859-2,
ASCII, ISO-646,
850,
1252,
1255,
SHIFT-JIS, MS-KANJI,
BIG5, FIVE,
UTF-7,
UTF-8,
UTF-16

Errors
1400 The document object is no longer valid.
1407 Invalid address for the return parameter was specified.
1416 Operation not supported in current view mode.

25.3.2.10.20 EndChanges

Method: EndChanges()

Description
Use the method EndChanges to display all changes since the call to Document.StartChanges 1244.

Errors
1400 The object is no longer valid.

25.3.2.10.21 ExecuteXQuery

Method: ExecuteXQuery (*strXMLFileName* as String)

Description

Execute the XQuery statements contained in the document of the document object. Either an XQuery execution or an XQuery Update is performed depending on the file extension of the document. Use the XML file specified in the argument as the XML target document that the XQuery document processes.

- If the document has an XQuery file extension as defined in the Options dialog of XMLSpy, then an XQuery execution is performed. By default: `.xq`, `.xql`, and `.xquery` are set as XQuery file extensions in XMLSpy.
- If the document has an XQuery Update file extension as defined in the Options dialog of XMLSpy, then an XQuery Update action is performed. By default: `.xqu` is set as an XQuery Update file extension in XMLSpy.

If your XQuery script does not use an XML source, set the parameter `strXMLFileName` to an empty string.

Errors

1400 The document object is no longer valid.
1423 XQuery transformation error.
1424 Not all files required for operation could be loaded. Most likely, the file specified in `strXMLFileName` does not exist or is not valid.

25.3.2.10.22 ExportToDatabase

Method: ExportToDatabase (*pFromChild* as XMLData[1309], *pExportSettings* as ExportSettings[1257], *pDatabase* as DatabaseConnection[1203])

Description

ExportToDatabase exports the XML document starting with the element pFromChild. The parameter pExportSettings defines the behaviour of the export (see Application.GetExportSettings[1139]). The parameter pDatabase specifies the destination of the export (see Application.GetDatabaseSettings[1138]). `UpdateXMLData()`[1246] might be indirectly needed as you have to pass the `XMLData`[1309] as parameter to this function.

Errors

1400 The object is no longer valid.
1407 Invalid parameter or invalid address for the return parameter was specified.
1416 Error during export.
1429 Database selection missing.
1430 Document export failed.

Example

```
Dim objDoc As Document
Set objDoc = objSpy.ActiveDocument

'set the behaviour of the export with ExportSettings
Dim objExpSettings As ExportSettings
Set objExpSettings = objSpy.GetExportSettings

'set the destination with DatabaseConnection
Dim objDB As DatabaseConnection
```

```
Set objDB = objSpy.GetDatabaseSettings

objDB.CreateMissingTables = True
objDB.CreateNew = True
objDB.File = "C:\Export.mdb"

objDoc.ExportToDatabase objDoc.RootElement, objExpSettings, objDB
If Err.Number <> 0 Then
        a = MsgBox("Error: " & (Err.Number - vbObjectError) & Chr(13) &
                        "Description: " & Err.Description)
End If
```

25.3.2.10.23 ExportToText

Method: ExportToText (*pFromChild* as XMLData[1309], *pExportSettings* as ExportSettings[1257], *pTextSettings* as TextImportExportSettings[1301])

Description
ExportToText exports tabular information from the document starting at pFromChild into one or many text files. Columns of the resulting tables are generated in alphabetical order of the column header names. Use GetExportElementList[1235] to learn about the data that will be exported. The parameter pExportSettings defines the specifics for the export. Set the property ExportSettings.ElementList[1258] to the - possibly modified - list returned by GetExportElementList[1235] to avoid exporting all contained tables. The parameter pTextSettings defines the options specific to text export and import. You need to set the property TextImportExportSettings.DestinationFolder[1302] before you call ExportToText. `UpdateXMLData()`[1246] might be indirectly needed as you have to pass the `XMLData`[1309] as parameter to this function.

See also Import and export of data[1103].

Errors

1400	The object is no longer valid.
1407	Invalid parameter or invalid address for the return parameter was specified.
1416	Error during export.
1430	Document export failed.

Example

```
' -------------------------------------------------------
' VBA client code fragment - export document to text files
' -------------------------------------------------------
        Dim objDoc As Document
        Set objDoc = objSpy.ActiveDocument

        Dim objExpSettings As ExportSettings
        Set objExpSettings = objSpy.GetExportSettings
        objExpSettings.ElementList = objDoc.GetExportElementList(
                                                        objDoc.RootElement,
                                                        objExpSettings)

        Dim objTextExp As TextImportExportSettings
        Set objTextExp = objSpy.GetTextImportExportSettings
        objTextExp.HeaderRow = True
        objTextExp.DestinationFolder = "C:\Exports"
```

```
On Error Resume Next
objDoc.ExportToText objDoc.RootElement, objExpSettings, objTextExp

If Err.Number <> 0 Then
        a = MsgBox("Error: " & (Err.Number - vbObjectError) & Chr(13) & "Description: "
        & Err.Description)
End If
```

25.3.2.10.24 FlattenDTDOrSchema

Method: FlattenDTDOrSchema (sOutputPath as String, nOutputPathDialogAction as SPYDialogAction[1361])

Parameters

`sOutputPath`
The file path for the newly generated file.

`nOutputPathDialogAction`
Defines the dialog interaction for this call.

Description

`FlattenDTDOrSchema` takes an existing DTD or schema, generates a flattened file, and saves the generated file at the specified location. In the case of DTDs, flattening removes parameter entities and produces a single DTD from a collection of modules; sections marked `IGNORE` are suppressed and unused parameter entities are deleted. When an XML Schema is flattened, (i) the components of all included schemas are added as global components of the active schema, and (ii) included schemas are deleted.

Errors

1400 The object is no longer valid.
1412 Error during conversion.

25.3.2.10.25 FullName

Property: FullName as String

Description

This property can be used to get or set the full file name - including the path - to where the document gets saved. The validity of the name is not verified before the next save operation.

This property makes the methods GetPathName[1236] and SetPathName[1243] obsolete.

Errors

1400 The document object is no longer valid.
1402 Empty string has been specified as full file name.

(C) 2015-2021 Altova GmbH

25.3.2.10.26 GenerateDTDOrSchema

Method: GenerateDTDOrSchema (*nFormat* as SPYDTDSchemaFormat[1361], *nValuesList* as integer, *nDetection* as SPYTypeDetection[1369], *nFrequentElements* as SPYFrequentElements[1362])

Parameters
nFormat
Sets the schema output format to DTD, or W3C.

nValuesList
Generate not more than this amount of enumeration-facets per type. Set to -1 for unlimited.

nDetection
Specifies granularity of simple type detection.

nFrequentElements
Shall the types for all elements be defined as global? Use the value *spyGlobalComplexType* to define them on global scope. Otherwise, use the value *spyGlobalElements*.

Description
Use this method to automatically generate a DTD or schema for the current XML document.
For a finer tuning of DTD / schema generation, use GenerateDTDOrSchemaEx[1233].
Note that this functionality is not available in ZIP View only. See Document.CurrentViewMode[1228]. and SPYViewModes[1370].

Errors
1400 The object is no longer valid.
1407 Invalid parameter or invalid address for the return parameter was specified.

25.3.2.10.27 GenerateDTDOrSchemaEx

Method: GenerateDTDOrSchemaEx (*objDlg* as DTDSchemaGeneratorDlg[1251]) as Document[1218]

Description
Use this method to automatically generate a DTD or schema for the current XML document. A DTDSchemaGeneratorDlg[1251] object is used to pass information to the schema/DTD generator. The generation process can be configured to allow user interaction or run without further user input.
Note that this functionality is not available in ZIP View only. See Document.CurrentViewMode[1228]. and SPYViewModes[1370].

Errors
1400 The object is no longer valid.
1407 Invalid parameter or invalid address for the return parameter was specified.

25.3.2.10.28 GenerateProgramCode

Method: GenerateProgramCode (*objDlg* as CodeGeneratorDlg[1203])
Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

Description
Generate Java, C++ or C# class files from the XML Schema definitions in your document. A CodeGeneratorDlg[1203] object is used to pass information to the code generator. The generation process can be configured to allow user interaction or run without further user input.

Errors
1400 The document object is no longer valid.
1407 An empty file name has been specified.
1421 Feature not available in this edition

25.3.2.10.29 GenerateSampleXML

Method: GenerateSampleXML (*objDlg* as GenerateSampleXMLDlg[1273]) as Document[1218]

Description
Generates a sample XML if the document is a schema or DTD. Use Dialogs.GenerateSampleXMLDlg[1217] to get an initialized set of options.

Available with TypeLibrary version 1.5

Errors
1400 The document object is no longer valid.

25.3.2.10.30 GenerateSchemaDocumentation

Method: GenerateSchemaDocumentation (*objDlg* as SchemaDocumentationDlg[1282])

Description
Generate documentation for a schema definition file in HTML, MS-Word, or RTF format. The parameter objDlg is used to parameterize the generation process. Use Dialogs.SchemaDocumentationDlg[1217] to get an initialized set of options. As a minimum, you will need to set the property SchemaDocumentationDlg.OutputFile[1290] before starting the generation process. Note that this functionality is limited to Schema View only. See Document.CurrentViewMode[1228] and SPYViewModes[1370].

Errors
1400 The document object is no longer valid.
1407 Invalid parameters have been passed or an empty file name has been specified as output target.
1417 The document is not opened in schema view, maybe it is not an '.xsd' file.
1421 Feature is not available in this edition.
1422 Error during generation

(C) 2015-2021 Altova GmbH

25.3.2.10.31 GetDBStructureList

Method: GetDBStructureList (*pDatabase* as DatabaseConnection[1209]) as ElementList[1255]

Description
GetDBStructureList creates a collection of elements from the Schema document for which tables in the specified database are created. The function returns a collection of ElementListItems where the properties ElementListItem.Name[1257] contain the names of the tables.

See also CreateDBStructureFromXMLSchema[1227].

Errors

1400	The object is no longer valid.
1427	Failed creating parser for the specified XML.
1428	Export of element list failed.
1429	Database selection missing.

25.3.2.10.32 GetExportElementList

Method: GetExportElementList (*pFromChild* as XMLData[1309], *pExportSettings* as ExportSettings[1257]) as ElementList[1255]

Description
GetExportElementList creates a collection of elements to export from the document, depending on the settings in pExportSettings and starting from the element pFromChild. The function returns a collection of ElementListItems where the properties ElementListItem.Name[1257] contain the names of the tables that can be exported from the document. The property ElementListItem.FieldCount[1257] contains the number of columns in the table. The property ElementListItem.RecordCount[1257] contains the number of records in the table. The property ElementListItem.ElementKind[1256] is unused. `UpdateXMLData()`[1246] might be indirectly needed as you have to pass the `XMLData`[1309] as parameter to this function.

See also Import and export of data[1103].

Errors

1400	The object is no longer valid.
1407	Invalid parameter or invalid address for the return parameter was specified.
1427	Failed creating parser for the specified XML.
1428	Export of element list failed.

25.3.2.10.33 GetPathName (obsolete)

Superseded by Document.FullName 1232

```
// ----- javascript sample -----
// instead of:
// strPathName = Application.ActiveDocument.GetPathName();
// use now:
strPathName = Application.ActiveDocument.FullName;
```

Method: GetPathName() as String

Description
The method GetPathName gets the path of the active document.

See also Document.SetPathName 1243 (obsolete).

25.3.2.10.34 GridView

Property: GridView as GridView 1279

Description
This property provides access to the grid view functionality of the document.

Errors

1400	The object is no longer valid.
1407	Invalid address for the return parameter was specified.
1417	Document needs to be open in enhanced grid view mode.

25.3.2.10.35 IsModified

Property: IsModified as Boolean

Description
True if the document is modified.

Errors

1400	The object is no longer valid.
1407	Invalid address for the return parameter was specified.

25.3.2.10.36 IsValid

Method: HRESULT IsValid([in, out] VARIANT *strError, [in, out] VARIANT *nErrorPos, [in, out] VARIANT *pBadData, [out,retval] VARIANT_BOOL *bValid);

(C) 2015-2021 Altova GmbH

Return Value

True if the document is valid, false if not. To call `IsValid()`, the application GUI must be visible. (If you wish to validate without the GUI being visible, please use Altova RaptorXML Server.)

Description

`IsValid` validates the document against its associated schema or DTD. `strError` gives you the same error message as when you validate the file within the GUI.

Errors

1400 The object is no longer valid.
1407 Invalid parameter or invalid address for the return parameter was specified.
1408 Unable to validate file.

Example

The following C++ code snippet provides an example of how to use the **`IsValid`** method.

```
#import "XMLSpy.tlb"


CComPtr< XMLSpyLib::IDocument12> ipDoc = ipXMLSpy->GetActiveDocument();
if ( ipDoc )
{
	// prepare in/out parameters for IsValid call
	CComVariant variantError;
	CComVariant variantErrorPos;
	CComVariant variantBadData;
	// IsValid always shows a dialog with the validation result. This cannot be turned
off.
	bool bIsValid = ipDoc->IsValid( &variantError, &variantErrorPos, &variantBadData )
== VARIANT_TRUE;

	if ( !bIsValid )
	{
		// retrieve values from out parameters
		CString strError = (V_VT( &variantError ) == VT_BSTR ?
V_BSTR( &variantError ) : _T( "" ));
		long npos = (V_VT( &variantErrorPos ) == VT_I4 ? V_I4( &variantErrorPos ) :
-1);
		CComQIPtr< XMLSpyLib::IXMLData > ipXMLBadData = (V_VT( &variantBadData ) ==
VT_DISPATCH ? V_DISPATCH( &variantBadData ) : nullptr);

		if ( ipXMLBadData )
			strError += CString( _T("\n\n Node: ") ) + (LPCWSTR)ipXMLBadData-
>GetName();

		if ( !strError.IsEmpty() )
			AfxMessageBox( "Validation failed - " + strError );
	}
}
```

25.3.2.10.37 IsValidEx

Method: IsValidEx (*nXSDVersion* as SPYValidateXSDVersion[1370], *nErrorLimit* as int, *nErrorFormat* as SPYValidateErrorFormat[1370], out *strError* as Variant) as Boolean

Return Value
True if the document is valid, false if not.

Description
IsValidEx validates the document against its associated schema or DTD.

In parameters:
nXSDVersion which is an enumeration value of SPYValidateXSDVersion[1370] that selects the XSD version to validate against.
nErrorLimit which is an integer. Values must be 1 to 999.
nErrorFormat which is an enumeration value of SPYValidateErrorFormat[1370] that selects the XSD version to validate against.

Out parameter:
strError is the error message, and is the same as that received when validating the file within the GUI.

Errors

1400	The object is no longer valid.
1407	Invalid parameter or invalid address for the return parameter was specified.
1408	Unable to validate file.

Example
The following C++ code snippet provides an example of how to use the **IsValidEx** method.

```
#import "XMLSpy.tlb"


CComPtr< XMLSpyLib::IDocument12> ipDoc = ipXMLSpy->GetActiveDocument();
if ( ipDoc )
{
	CComVariant variantErrorEx;
	bool bIsValidEx = ipDoc->IsValidEx( XMLSpyLib::spyValidateXSDVersion_AutoDetect,
100, XMLSpyLib::SPYValidateErrorFormat_LongXML, &variantErrorEx ) == VARIANT_TRUE;

	//	// retrieve values from out parameters
	CString strErrorEx = (V_VT( &variantErrorEx ) == VT_BSTR ?
V_BSTR( &variantErrorEx ) : _T( "" ));
	if ( !strErrorEx.IsEmpty() )
		AfxMessageBox( "Validation failed - " + strErrorEx );
}
```

25.3.2.10.38 IsWellFormed

Method: IsWellFormed (*pData* as XMLData, *bWithChildren* as Boolean, *strError* as Variant, *nErrorPos* as Variant, *pBadXMLData* as Variant) as Boolean

Return Value
True if the document is well formed.

Description
IsWellFormed checks the document for well-formedness starting at the element pData.

If the document is not well formed, strError contains an error message, nErrorPos the position in the file and pBadXMLData holds a reference to the element which breaks the well-formedness. These out-parameters are defined as VARIANTs to support scripting languages like VBScript.

Errors
1400 The object is no longer valid.
1407 Invalid parameter or invalid address for the return parameter was specified.

Example

See IsValid.

25.3.2.10.39 Name

Property: Name as String (read-only)

Description
Use this property to retrieve the name - not including the path - of the document file. To change the file name for a document use the property FullName 1232.

Errors
1400 The document object is no longer valid.
1407 Invalid address for the return parameter was specified.

25.3.2.10.40 Parent

Property: Parent as Documents 1246 (read-only)

Description
Access the parent of the document object.

Errors
1400 The document object is no longer valid.
1407 Invalid address for the return parameter was specified.

Property: Parent as Application[1131] (read-only)

25.3.2.10.41 Path

Property: Path as String (read-only)

Description
Use this property to retrieve the path - not including the file name - of the document file. To change the file name and path for a document use the property FullName[1232].

Errors
1400 The document object is no longer valid.
1407 Invalid address for the return parameter was specified.

25.3.2.10.42 RootElement

Property: RootElement as XMLData[1309] (read-only)

Description
The property RootElement provides access to the root element of the XML structure of the document including the XML prolog data. To access the first element of a document's content navigate to the first child of kind *spyXMLDataElement* or use the Document.DataRoot[1228] property. If the CurrentViewMode[1228] is not *spyViewGrid* or *spyViewAuthentic* an UpdateXMLData[1246] may be necessary to get access to the latest XMLData[1309].

Errors
1400 The document object is no longer valid.
1407 Invalid address for the return parameter was specified.

25.3.2.10.43 Save

Method: Save()

Description
The method writes any modifications of the document to the associated file. See also Document.FullName[1232].

Errors
1400 The document object is no longer valid.
1407 An empty file name has been specified.
1403 Error when saving file, probably the file name is invalid.

(C) 2015-2021 Altova GmbH

25.3.2.10.44 SaveAs

Method: SaveAs (*strFileName* as String)

Description
Save the document to the file specified. If saving was successful, the FullName[1232] property gets set to the specified file name.

Errors
1400 The document object is no longer valid.
1407 An empty file name has been specified.
1403 Error when saving file, probably the file name is invalid.

25.3.2.10.45 Saved

Property: Saved as Boolean (read-only)

Description
This property can be used to check if the document has been saved after the last modifications. It returns the negation of IsModified[1236].

Errors
1400 The document object is no longer valid.
1407 Invalid address for the return parameter was specified.

25.3.2.10.46 SaveInString

Method: SaveInString (*pData* as XMLData[1309], *bMarked* as Boolean) as String

Parameters
pData
XMLData element to start. Set pData to Document.RootElement[1240] if you want to copy the complete file.

bMarked
If bMarked is true, only the elements selected in the grid view are copied.

Return Value
Returns a string with the XML data.

Description
SaveInString starts at the element pData and converts the XMLData objects to a string representation. `UpdateXMLData()`[1246] might be indirectly needed as you have to pass the `XMLData`[1309] as parameter to this function.

Errors
1400 The object is no longer valid.
1407 Invalid parameter or invalid address for the return parameter was specified.

25.3.2.10.47 SaveToURL

Method: SaveToURL (*strURL* as String, *strUser* as String, *strPassword* as String)

Return Value

Description
SaveToURL() writes the document to the URL strURL. This method does not set the permanent file path of the document.

Errors
1400 The object is no longer valid.
1402 Invalid URL specified.
1403 Error while saving to URL.

25.3.2.10.48 SetActiveDocument

Method: SetActiveDocument()

Description
The method sets the document as the active and brings it to the front.

Errors
1400 The object is no longer valid.

25.3.2.10.49 SetEncoding (obsolete)

Superseded by Document.Encoding 1229

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.SetEncoding("UTF-16");
// use now:
Application.ActiveDocument.Encoding = "UTF-16";
```

Method: SetEncoding (*strEncoding* as String)

Description
SetEncoding sets the encoding of the document like the menu item "File/Encoding..." in XMLSpy. Possible values for strEncoding are, for example:

8859-1,
8859-2,
ASCII, ISO-646,

 (C) 2015-2021 Altova GmbH

850,
1252,
1255,
SHIFT-JIS, MS-KANJI,
BIG5, FIVE,
UTF-7,
UTF-8,
UTF-16

25.3.2.10.50 SetExternalIsValid

Method: SetExternalIsValid (*bValid* as Boolean)

Parameters

bValid
Sets the result of an external validation process.

Description
The internal information set by this method is only queried on cancelling the default validation in any OnBeforeValidate[1221] handler.

Available with TypeLibrary version 1.5

Errors
1400 The object is no longer valid.

25.3.2.10.51 SetPathName (obsolete)

Superseded by Document.FullName[1232]

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.SetPathName("C:\\myXMLFiles\\test.xml");
// use now:
Application.ActiveDocument.FullName = "C:\\myXMLFiles\\test.xml";
```

Method: SetPathName (*strPath* as String)

Description
The method SetPathName sets the path of the active document. SetPathName only copies the string and does not check if the path is valid. All succeeding save operations are done into this file.

25.3.2.10.52 StartChanges

Method: StartChanges()

Description
After StartChanges is executed XMLSpy will not update its editor windows until Document.EndChanges 1229 is called. This increases performance of complex tasks to the XML structure.

Errors
1400 The object is no longer valid.

25.3.2.10.53 Suggestions

Property: Suggestions as Array

Description
This property contains the last valid user suggestions for this document. The XMLSpy generated suggestions can be modified before they are shown to the user in the OnBeforeShowSuggestions 1304 event.

Errors
1400 The object is no longer valid.
1407 Invalid parameter or invalid address for the return parameter was specified.

25.3.2.10.54 SwitchViewMode

Method: SwitchViewMode (*nMode* as SPYViewModes 1370) as Boolean

Return value
Returns true if view mode is switched.

Description
The method sets the current view mode of the document in XMLSpy. See also Document.CurrentViewMode 1228 .

Errors
1400 The object is no longer valid.
1407 Invalid address for the return parameter was specified.
1417 Invalid view mode specified.

25.3.2.10.55 TextView

Property: TextView as TextView 1304

Description
This property provides access to the text view functionality of the document.

(C) 2015-2021 Altova GmbH

Errors
1400 The object is no longer valid.
1407 Invalid address for the return parameter was specified.

25.3.2.10.56 Title

Property: Title as String (read-only)

Description
Title contains the file name of the document. To get the path and filename of the file use FullName 1232.

Errors
1400 The document object is no longer valid.
1407 Invalid address for the return parameter was specified.

25.3.2.10.57 TransformXSL

Method: TransformXSL()

Description
TransformXSL processes the XML document via the associated XSL file. See Document.AssignXSL 1223 on how to place a reference to a XSL file into the document.

Errors
1400 The document object is no longer valid.
1411 Error during transformation process.

25.3.2.10.58 TransformXSLEx

Method: TransformXSLEx(*nAction* as SPYDialogAction 1361)

Description
TransformXSLEx processes the XML document via the associated XSL file. The parameter specifies whether a dialog asking for the result document name should pop up or not. See Document.AssignXSL 1223 on how to place a reference to a XSL file into the document.

Errors
1400 The document object is no longer valid.
1411 Error during transformation process.

25.3.2.10.59 TransformXSLFO

Method: TransformXSLFO()

Description
TransformXSLFO processes the XML document via the associated XSLFO file. See AssignXSLFO[1223] on how to place a reference to a XSLFO file into the document. You need to assign a FOP processor to XMLSpy before you can use this method.

Errors
1400 The document object is no longer valid.
1411 Error during transformation process.

25.3.2.10.60 UpdateViews

Method: UpdateViews()

Description
To redraw the Enhanced Grid View and the Tree View call UpdateViews. This can be important after you changed the XMLData structure of a document. This method does not redraw the text view of XMLSpy.

Errors
1400 The document object is no longer valid.

25.3.2.10.61 UpdateXMLData

Method: UpdateXMLData() as Boolean

Description
The XMLData[1309] tree is updated from the current view. Please note that this can fail in case of the TextView if the current XML text is not well-formed. This is not necessary if CurrentViewMode[1228] is *spyViewGrid* or *spyViewAuthentic* because these views keep the XMLData[1309] updated.

Available with TypeLibrary version 1.5

Errors
1400 The document object is no longer valid.

25.3.2.11 Documents

Properties
Count[1247]
Item[1248]

Methods
NewAuthenticFile[1248]
NewFile[1248]

NewFileFromText 1249
OpenAuthenticFile 1249
OpenFile 1249
OpenURL 1250
OpenURLDialog 1250

Description
This object represents the set of documents currently open in XMLSpy. Use this object to open further documents or iterate through already opened documents.

Examples

```
' ----------------------------------------
' XMLSpy scripting environment - VBScript
' iterate through open documents
' ----------------------------------------
Dim objDocuments
Set objDocuments = Application.Documents

For Each objDoc In objDocuments
        'do something useful with your document
        objDoc.SetActiveDocument()
Next
```

```
// ----------------------------------------
// XMLSpy scripting environment - JScript
// close all open documents
// ----------------------------------------
for (var iter = new Enumerator (Application.Documents);
    ! iter.atEnd();
    iter.moveNext())
{
        // MsgBox ("Closing file " + iter.item().Name);
        iter.item().Close (true);
}
```

25.3.2.11.1 Count

Property: Count as long

Description
Count of open documents.

Errors

1600	Invalid `Documents` object
1601	Invalid input parameter

25.3.2.11.2 Item

Method: Item (*n* as long) as Document[1218]

Description
Gets the document with the index n in this collection. Index is 1-based.

Errors
1600 Invalid `Documents` object
1601 Invalid input parameter

25.3.2.11.3 NewAuthenticFile

Method: NewAuthenticFile (*strSPSPath* as String, *strXMLPath* as String) as Document[1218]

Parameters
strSPSPath
The path to the SPS document.

strXMLPath
The new XML document name.

Return Value
The method returns the new document.

Description
NewAuthenticFile creates a new XML file and opens it in Authentic View using SPS design strSPSPath.

25.3.2.11.4 NewFile

Method: NewFile (*strFile* as String, *strType* as String) as Document[1218]

Parameters
strFile
Full path of new file.

strType
Type of new file as string (i.e. "xml", "xsd", ...)

Return Value
Returns the new file.

Description
NewFile creates a new file of type strType (i.e. "xml"). The newly created file is also the ActiveDocument.

(C) 2015-2021 Altova GmbH

25.3.2.11.5 NewFileFromText

Method: NewFileFromText (*strText* as String, *strType* as String) as Document[1218]

Parameters
strText
The content of the new document in plain text.

strType
Type of the document to create (i.e. "xml").

Return Value
The method returns the new document.

Description
NewFileFromText creates a new document with strText as its content.

25.3.2.11.6 OpenAuthenticFile

Method: OpenAuthenticFile (*strSPSPath* as String, *strXMLPath* as String) as Document[1218]

Parameters
strSPSPath
The path to the SPS document.

strXMLPath
The path to the XML document (can be empty).

Return Value
The method returns the new document.

Description
OpenAuthenticFile opens an XML file or database in Authentic View using SPS design strSPSPath.

25.3.2.11.7 OpenFile

Method: OpenFile (*strPath* as String, *bDialog* as Boolean) as Document[1218]

Parameters
strPath
Path and file name of file to open.

bDialog
Show dialogs for user input.

Return Value
Returns the opened file on success.

Description
OpenFile opens the file strPath. If bDialog is TRUE, a file-dialog will be displayed.

Example

```
Dim objDoc As Document
Set objDoc = objSpy.Documents.OpenFile(strFile, False)
```

25.3.2.11.8 OpenURL

Method: OpenURL (*strURL* as String, *nURLType* as SPYURLTypes[1369], *nLoading* as SPYLoading[1364], *strUser* as String, *strPassword* as String) as Document[1218]

Parameters
strURL
URL to open as document.

nURLType
Type of document to open. Set to -1 for auto detection.

nLoading
Set nLoading to 0 (zero) if you want to load it from cache or proxy. Otherwise set nLoading to 1.

strUser
Name of the user if required. Can be empty.

strPassword
Password for authentification. Can be empty.

Return Value
The method returns the opened document.

Description
OpenURL opens the URL strURL.

25.3.2.11.9 OpenURLDialog

Method: OpenURLDialog (*strURL* as String, *nURLType* as SPYURLTypes[1369], *nLoading* as SPYLoading[1364], *strUser* as String, *strPassword* as String) as Document[1218]

Parameters
strURL
URL to open as document.

nURLType
Type of document to open. Set to -1 for auto detection.

nLoading
Set nLoading to 0 (zero) if you want to load it from cache or proxy. Otherwise set nLoading to 1.

strUser
Name of the user if required. Can be empty.

strPassword

(C) 2015-2021 Altova GmbH

Password for authentification. Can be empty.

Return Value
The method returns the opened document.

Description
OpenURLDialog displays the "open URL" dialog to the user and presets the input fields with the given parameters.

25.3.2.12 DTDSchemaGeneratorDlg

Properties and Methods

Standard automation properties
Application 1251
Parent 1254

DTDSchemaFormat 1252
ValueList 1255
TypeDetection 1255
FrequentElements 1252
MergeAllEqualNamed 1253
ResolveEntities 1254
AttributeTypeDefinition 1252
GlobalAttributes 1252
OnlyStringEnums 1253
MaxEnumLength 1253
OutputPath 1253
OutputPathDialogAction 1254

Description
Use this object to configure the generation of a schema or DTD. The method GenerateDTDOrSchemaEx 1233 expects a DTDSchemaGeneratorDlg as parameter to configure the generation as well as the associated user interactions.

25.3.2.12.1 Application

Property: Application as Application 1131 (read-only)

Description
Access the XMLSpy application object.

Errors
3000 The object is no longer valid.
3001 Invalid address for the return parameter was specified.

25.3.2.12.2 AttributeTypeDefinition

Property: AttributeTypeDefinition as SPYAttributeTypeDefinition 1358

Description
Specifies how attribute definitions get merged.

Errors
3000 The object is no longer valid.
3001 Invalid address for the return parameter was specified.

25.3.2.12.3 DTDSchemaFormat

Property: DTDSchemaFormat as SPYDTDSchemaFormat 1361

Description
Sets the schema output format to DTD, or W3C.

Errors
3000 The object is no longer valid.
3001 Invalid address for the return parameter was specified.

25.3.2.12.4 FrequentElements

Property: FrequentElements as SPYFrequentElements 1362

Description
Shall the types for all elements be defined as global? Use the value *spyGlobalComplexType* to define them on global scope. Otherwise, use the value *spyGlobalElements*.

Errors
3000 The object is no longer valid.
3001 Invalid address for the return parameter was specified.

25.3.2.12.5 GlobalAttributes

Property: GlobalAttributes as Boolean

Description
Shall attributes with same name and type be resolved globally?

Errors
3000 The object is no longer valid.

(C) 2015-2021 Altova GmbH

3001 Invalid address for the return parameter was specified.

25.3.2.12.6 MaxEnumLength

Property: MaxEnumLength as Integer

Description
Specifies the maximum number of characters allowed for enumeration names. If one value is longer than this, no enumeration will be generated.

Errors
3000 The object is no longer valid.
3001 Invalid address for the return parameter was specified.

25.3.2.12.7 MergeAllEqualNamed

Property: MergeAllEqualNamed as Boolean

Description
Shall types of all elements with the same name be merged into one type?

Errors
3000 The object is no longer valid.
3001 Invalid address for the return parameter was specified.

25.3.2.12.8 OnlyStringEnums

Property: OnlyStringEnums as Boolean

Description
Specifies if enumerations will be created only for plain strings or all types of values.

Errors
3000 The object is no longer valid.
3001 Invalid address for the return parameter was specified.

25.3.2.12.9 OutputPath

Property: OutputPath as String

Description

Selects the file name for the generated schema/DTD.

Errors
3000 The object is no longer valid.
3001 Invalid address for the return parameter was specified.

25.3.2.12.10 OutputPathDialogAction

Property: OutputPathDialogAction as SPYDialogAction 1361

Description
Defines how the sub-dialog for selecting the schema/DTD output path gets handled. Set this value to *spyDialogUserInput(2)* to show the dialog with the current value of the OutputPath 1253 property as default. Use *spyDialogOK(0)* to hide the dialog from the user.

Errors
3000 The object is no longer valid.
3001 Invalid address for the return parameter was specified.

25.3.2.12.11 Parent

Property: Parent as Dialogs 1215 (read-only)

Description
Access the parent of the object.

Errors
3000 The object is no longer valid.
3001 Invalid address for the return parameter was specified.

25.3.2.12.12 ResolveEntities

Property: ResolveEntities as Boolean

Description
Shall all entities be resolved before generation starts? If yes, an info-set will be built.

Errors
3000 The object is no longer valid.
3001 Invalid address for the return parameter was specified.

(C) 2015-2021 Altova GmbH

25.3.2.12.13 TypeDetection

Property: TypeDetection as SPYTypeDetection 1369

Description
Specifies granularity of simple type detection.

Errors
3000 The object is no longer valid.
3001 Invalid address for the return parameter was specified.

25.3.2.12.14 ValueList

Property: ValueList as Integer

Description
Generate not more than this amount of enumeration-facets per type. Set to -1 for unlimited.

Errors
3000 The object is no longer valid.
3001 Invalid address for the return parameter was specified.

25.3.2.13 ElementList

Properties
Count 1256
Item 1256

Methods
RemoveElement 1256

Description
Element lists are used for different purposes during export and import of data. Depending on this purpose, different properties of ElementListItem 1256 are used.

It can hold
- a list of table names returned by a call to Application.GetDatabaseTables 1138,
- a list of field names retuned by a call to Application.GetDatabaseImportElementList 1137 or Application.GetTextImportElementList 1139,
- a field name filter list used in Application.ImportFromDatabase 1141 and Application.ImportFromText 1143,
- a list of table names and counts for their rows and columns as returned by calls to GetExportElementList 1235 or
- a field name filter list used in Document.ExportToDatabase 1230 and Document.ExportToText 1231.

25.3.2.13.1 Count

Property: Count as long (read-only)

Description
Count of elements in this collection.

25.3.2.13.2 Item

Method: Item(n as long) as ElementListItem 1256

Description
Gets the element with the index n from this collection. The first item has index 1.

25.3.2.13.3 RemoveElement

Method: RemoveElement(Index as long)

Description
RemoveElement removes the element Index from the collection. The first Item has index 1.

25.3.2.14 ElementListItem

Properties
Name 1257

ElementKind 1256

FieldCount 1257
RecordCount 1257

Description
An element in an ElementList 1255. Usage of its properties depends on the purpose of the element list. For details see ElementList 1255.

25.3.2.14.1 ElementKind

Property: ElementKind as SPYXMLDataKind 1371

Description
Specifies if a field should be imported as XML element (data value of spyXMLDataElement) or attribute (data value of spyXMLDataAttr).

(C) 2015-2021 Altova GmbH

25.3.2.14.2 FieldCount

Property: FieldCount as long (read-only)

Description
Count of fields (i.e. columns) in the table described by this element. This property is only valid after a call to Document.GetExportElementList[1235].

25.3.2.14.3 Name

Property: Name as String (read-only)

Description
Name of the element. This is either the name of a table or a field, depending on the purpose of the element list.

25.3.2.14.4 RecordCount

Property: RecordCount as long (read-only)

Description
Count of records (i.e. rows) in the table described by this element. This property is only valid after a call to Document.GetExportElementList[1235].

25.3.2.15 ExportSettings

Properties

ElementList[1258]

EntitiesToText[1258]

ExportAllElements[1258]
SubLevelLimit[1260]

FromAttributes[1259]
FromSingleSubElements[1259]
FromTextValues[1259]

CreateKeys[1258]
IndependentPrimaryKey[1259]

Namespace[1259]

ExportCompleteXML[1258]
StartFromElement[1259]

Description

ExportSettings contains options used during export of XML data to a database or text file. See Import and export of data 1103 for a general overview.

25.3.2.15.1 CreateKeys

Property: CreateKeys as Boolean

Description
This property turns creation of keys (i.e. primary key and foreign key) on or off. Default is True.

25.3.2.15.2 ElementList

Property: ElementList as ElementList 1255

Description
Default is empty list. This list of elements defines which fields will be exported. To get the list of available fields use Document.GetExportElementList 1235. It is possible to prevent exporting columns by removing elements from this list with ElementList.RemoveElement 1256 before passing it to Document.ExportToDatabase 1230 or Document.ExportToText 1231.

25.3.2.15.3 EntitiesToText

Property: EntitiesToText as Boolean

Description
Defines if XML entities should be converted to text or left as they are during export. Default is True.

25.3.2.15.4 ExportAllElements

Property: ExportAllElements as Boolean

Description
If set to true, all elements in the document will be exported. If set to false, then ExportSettings.SubLevelLimit 1260 is used to restrict the number of sub levels to export. Default is true.

25.3.2.15.5 ExportCompleteXML

Property: ExportCompleteXML as Boolean

Description
Defines whether the complete XML is exported or only the element specified by StartFromElement 1259 and its children. Default is True.

(C) 2015-2021 Altova GmbH

25.3.2.15.6 FromAttributes

Property: FromAttributes as Boolean

Description
Set FromAttributes to false if no export data should be created from attributes. Default is True.

25.3.2.15.7 FromSingleSubElements

Property: FromSingleSubElements as Boolean

Description
Set FromSingleSubElements to false if no export data should be created from elements. Default is True.

25.3.2.15.8 FromTextValues

Property: FromTextValues as Boolean

Description
Set FromTextValues to false if no export data should be created from text values. Default is True.

25.3.2.15.9 IndependentPrimaryKey

Property: IndependentPrimaryKey as Boolean

Description
Turns creation of independent primary key counter for every element on or off. If ExportSettings.CreateKeys 1258 is False, this property will be ignored. Default is True.

25.3.2.15.10 Namespace

Property: Namespace as SPYExportNamespace 1362

Description
The default setting removes all namespace prefixes from the element names. In some database formats the colon is not a legal character. Default is spyNoNamespace.

25.3.2.15.11 StartFromElement

Property: StartFromElement as String

Description
Specifies the start element for the export. This property is only considered when ExportCompleteXML 1258 is false.

25.3.2.15.12 SubLevelLimit

Property: SubLevelLimit as Integer

Description
Defines the number of sub levels to include for the export. Default is 0. This property is ignored if ExportSettings.ExportAllElements[1258] is true.

25.3.2.16 FileSelectionDlg

Properties and Methods

Standard automation properties
Application[1260]
Parent[1261]

Dialog properties
FullName[1261]

Acceptance or cancellation of action that caused event
DialogAction[1260]

Description
The dialog object allows you to receive information about an event and pass back information to the event handler in the same way as with a user dialog. Use the FileSelectionDlg.FullName[1261] to select or modify the file path and set the FileSelectionDlg.DialogAction[1260] property to cancel or agree with the action that caused the event.

25.3.2.16.1 Application

Property: Application as Application[1131] (read-only)

Description
Access the XMLSpy application object.

Errors
2400 The object is no longer valid.
2401 Invalid address for the return parameter was specified.

25.3.2.16.2 DialogAction

Property: DialogAction as SPYDialogAction[1361]

Description
If you want your script to perform the file selection operation without any user interaction necessary, simulate user interaction by either setting the property to *spyDialogOK(0)* or *spyDialogCancel(1)*.

(C) 2015-2021 Altova GmbH

To allow your script to fill in the default values but let the user see and react on the dialog, use the value *spyDialogUserInput(2)*. If you receive a FileSelectionDlg object in an event handler, *spyDialogUserInput(2)* is not supported and will be interpreted as *spyDialogOK(0)*.

Errors
2400 The object is no longer valid.
2401 Invalid value for dialog action or invalid address for the return parameter was specified.

25.3.2.16.3 FullName

Property: FullName as String

Description
Access the full path of the file the gets selected by the dialog. Most events that pass a FileSelectionDlg object to you allow you modify this value and thus influence the action that caused the event (e.g. load or save to a different location).

Errors
2400 The object is no longer valid.
2401 Invalid address for the return parameter was specified.

25.3.2.16.4 Parent

Property: Parent as Dialogs 1215 (read-only)

Description
Access the parent of the object.

Errors
2400 The object is no longer valid.
2401 Invalid address for the return parameter was specified.

25.3.2.17 FindInFilesDlg

Properties and Methods

Standard automation properties
Application 1262
Parent 1264

Find 1263
RegularExpression 1264
Replace 1265
DoReplace 1263
ReplaceOnDisk 1265
MatchWholeWord 1264

MatchCase 1264
SearchLocation 1265
StartFolder 1266
IncludeSubfolders 1263
SearchInProjectFilesDoExternal 1265
FileExtension 1263
AdvancedXMLSearch 1262
XMLElementNames 1267
XMLElementContents 1267
XMLAttributeNames 1266
XMLAttributeContents 1266
XMLComments 1267
XMLCData 1267
XMLPI 1268
XMLRest 1268
ShowResult 1266

Description
Use this object to configure the search (or replacement) for strings in files. The method FindInFiles 1137 expects a FindInFilesDlg as parameter.

25.3.2.17.1 AdvancedXMLSearch

Property: AdvancedXMLSearch as Boolean

Description
Specifies if the XML search properties (XMLElementNames 1267, XMLElementContents 1267, XMLAttributeNames 1266, XMLAttributeContents 1267, XMLComments 1267, XMLCData 1267, XMLPI 1268 and XMLRest 1268) are considered. The default is false.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

25.3.2.17.2 Application

Property: Application as Application 1131 (read-only)

Description
Access the XMLSpy application object.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

(C) 2015-2021 Altova GmbH

25.3.2.17.3 DoReplace

Property: DoReplace as Boolean

Description
Specifies if the matched string is replaced by the string defined in Replace 1265. The default is false.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

25.3.2.17.4 FileExtension

Property: FileExtension as String

Description
Specifies the file filter of the files that should be considered during the search. Multiple file filters must be delimited with a semicolon (eg: *.xml;*.dtd;a*.xsd). Use the wildcards * and ? to define the file filter.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

25.3.2.17.5 Find

Property: Find as String

Description
Specifies the string to search for.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

25.3.2.17.6 IncludeSubfolders

Property: IncludeSubfolders as Boolean

Description
Specifies if subfolders are searched too. The default is true.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

25.3.2.17.7 MatchCase

Property: MatchCase as Boolean

Description
Specifies if the search is case sensitive. The default is true.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

25.3.2.17.8 MatchWholeWord

Property: MatchWholeWord as Boolean

Description
Specifies whether the whole word or just a part of it must match. The default is false.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

25.3.2.17.9 Parent

Property: Parent as Dialogs 1215 (read-only)

Description
Access the parent of the object.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

25.3.2.17.10 RegularExpression

Property: RegularExpression as Boolean

Description
Specifies if Find 1263 contains a regular expression. The default is false.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

(C) 2015-2021 Altova GmbH

25.3.2.17.11 Replace

Property: Replace as String

Description
Specifies the replacement string. The matched string is only replaced if DoReplace 1263 is set true.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

25.3.2.17.12 ReplaceOnDisk

Property: ReplaceOnDisk as Boolean

Description
Specifies if the replacement is done directly on disk. The modified file is not opened. The default is false.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

25.3.2.17.13 SearchInProjectFilesDoExternal

Property: SearchInProjectFilesDoExternal as Boolean

Description
Specifies if the external folders in the open project are searched, when a project search is performed. The default is false.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

25.3.2.17.14 SearchLocation

Property: SearchLocation as SPYFindInFilesSearchLocation 1362

Description
Specifies the location of the search. The default is spyFindInFiles_Documents.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

25.3.2.17.15 ShowResult

Property: ShowResult as Boolean

Description
Specifies if the result is displayed in the Find in Files output window. The default is false.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

25.3.2.17.16 StartFolder

Property: StartFolder as String

Description
Specifies the folder where the disk search starts.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

25.3.2.17.17 XMLAttributeContents

Property: XMLAttributeContents as Boolean

Description
Specifies if attribute contents are searched when AdvancedXMLSearch 1262 is true. The default is true.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

25.3.2.17.18 XMLAttributeNames

Property: XMLAttributeNames as Boolean

Description
Specifies if attribute names are searched when AdvancedXMLSearch 1262 is true. The default is true.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

(C) 2015-2021 Altova GmbH

25.3.2.17.19 XMLCData

Property: XMLCData as Boolean

Description
Specifies if CData tags are searched when AdvancedXMLSearch 1262 is true. The default is true.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

25.3.2.17.20 XMLComments

Property: XMLComments as Boolean

Description
Specifies if comments are searched when AdvancedXMLSearch 1262 is true. The default is true.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

25.3.2.17.21 XMLElementContents

Property: XMLElementContents as Boolean

Description
Specifies if element contents are searched when AdvancedXMLSearch 1262 is true. The default is true.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

25.3.2.17.22 XMLElementNames

Property: XMLElementNames as Boolean

Description
Specifies if element names are searched when AdvancedXMLSearch 1262 is true. The default is true.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

25.3.2.17.23 XMLPI

Property: XMLPI as Boolean

Description
Specifies if XML processing instructions are searched when AdvancedXMLSearch[1262] is true. The default is true.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

25.3.2.17.24 XMLRest

Property: XMLRest as Boolean

Description
Specifies if the rest of the XML (which is not covered by the other XML search properties) is searched when AdvancedXMLSearch[1262] is true. The default is true.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

25.3.2.18 FindInFilesResult

Properties and Methods

Standard automation properties
Application[1268]
Parent[1269]

Count[1269]
Item[1269]

Path[1270]
Document[1269]

Description
This object represents a file that matched the search criteria. It contains a list of FindInFilesResultMatch[1270] objects that describe the matching position.

25.3.2.18.1 Application

Property: Application as Application[1131] (read-only)

Description

(C) 2015-2021 Altova GmbH

Access the XMLSpy application object.

Errors
3700 The object is no longer valid.
3701 Invalid address for the return parameter was specified.

25.3.2.18.2 Count

Property: Count as long (read-only)

Description
Count of elements in this collection.

25.3.2.18.3 Document

Property: Path as Document 1218 (read-only)

Description
This property returns the Document 1218 object if the matched file is already open in XMLSpy.

Errors
3700 The object is no longer valid.
3701 Invalid address for the return parameter was specified.

25.3.2.18.4 Item

Method: Item(n as long) as FindInFilesResultMatch 1270

Description
Gets the element with the index n from this collection. The first item has index 1.

25.3.2.18.5 Parent

Property: Parent as FindInFilesResults 1272 (read-only)

Description
Access the parent of the object.

Errors
3700 The object is no longer valid.
3701 Invalid address for the return parameter was specified.

25.3.2.18.6 Path

Property: Path as String (read-only)

Description
Returns the path of the file that matched the search criteria.

Errors
3700 The object is no longer valid.
3701 Invalid address for the return parameter was specified.

25.3.2.19 FindInFilesResultMatch

Properties and Methods

Standard automation properties
Application 1270
Parent 1271

Line 1271
Position 1272
Length 1270
LineText 1271
Replaced 1272

Description
Contains the exact position in the file of the matched string.

25.3.2.19.1 Application

Property: Application as Application 1131 (read-only)

Description
Access the XMLSpy application object.

Errors
3800 The object is no longer valid.
3801 Invalid address for the return parameter was specified.

25.3.2.19.2 Length

Property: Length as Long (read-only)

Description
Returns the length of the matched string.

(C) 2015-2021 Altova GmbH

Errors
3800 The object is no longer valid.
3801 Invalid address for the return parameter was specified.

25.3.2.19.3 Line

Property: Line as Long (read-only)

Description
Returns the line number of the match. The line numbering starts with 0.

Errors
3800 The object is no longer valid.
3801 Invalid address for the return parameter was specified.

25.3.2.19.4 LineText

Property: LineText as String (read-only)

Description
Returns the text of the line.

Errors
3800 The object is no longer valid.
3801 Invalid address for the return parameter was specified.

25.3.2.19.5 Parent

Property: Parent as FindInFilesResult 1268 (read-only)

Description
Access the parent of the object.

Errors
3800 The object is no longer valid.
3801 Invalid address for the return parameter was specified.

25.3.2.19.6 Position

Property: Position as Long (read-only)

Description
Returns the start position of the match in the line. The position numbering starts with 0.

Errors
3800 The object is no longer valid.
3801 Invalid address for the return parameter was specified.

25.3.2.19.7 Replaced

Property: Replaced as Boolean (read-only)

Description
True if the matched string was replaced.

Errors
3800 The object is no longer valid.
3801 Invalid address for the return parameter was specified.

25.3.2.20 FindInFilesResults

Properties and Methods

Standard automation properties
Application 1272
Parent 1273

Count 1273
Item 1273

Description
This is the result of the FindInFiles 1137 method. It is a list of FindInFilesResult 1268 objects.

25.3.2.20.1 Application

Property: Application as Application 1131 (read-only)

Description
Access the XMLSpy application object.

Errors
3600 The object is no longer valid.

(C) 2015-2021 Altova GmbH

3601 Invalid address for the return parameter was specified.

25.3.2.20.2 Count

Property: Count as long (read-only)

Description
Count of elements in this collection.

25.3.2.20.3 Item

Method: Item(n as long) as FindInFilesResult 1268

Description
Gets the element with the index n from this collection. The first item has index 1.

25.3.2.20.4 Parent

Property: Parent as Application 1131 (read-only)

Description
Access the parent of the object.

Errors
3600 The object is no longer valid.
3601 Invalid address for the return parameter was specified.

25.3.2.21 GenerateSampleXMLDlg

Properties and Methods

Standard automation properties
Application 1274
Parent 1277

NonMandatoryAttributes 1276
NonMandatoryElements 1276
RepeatCount 1277
FillAttributesWithSampleData 1275
FillElementsWithSampleData 1275
ContentOfNillableElementsIsNonMandatory 1275
TryToUseNonAbstractTypes 1278
SchemaOrDTDAssignment 1278
LocalNameOfRootElement 1276
NamespaceURIOfRootElement 1276

OptionsDialogAction[1277]

Properties that are no longer supported
TakeFirstChoice - obsolete[1278]
FillWithSampleData - obsolete[1275]
Optimization - obsolete[1277]

Description
Used to set the parameters for the generation of sample XML instances based on a W3C schema or DTD.

25.3.2.21.1 Application

Property: Application as Application[1131] (read-only)

Description
Access the XMLSpy application object.

Errors
2200 The object is no longer valid.
2201 Invalid address for the return parameter was specified.

25.3.2.21.2 ChoiceMode

Property: `ChoiceMode` as SPYSampleXMLGenerationChoiceMode[1366]

Description
Specifies which elements will be generated.

Errors

2200 The object is no longer valid.
2201 Invalid address for the return parameter was specified.

25.3.2.21.3 ConsiderSampleValueHints

Property: `ConsiderSampleValueHints` as Boolean

Description
Selects whether to use SampleValueHints[1278] or not.

Errors

(C) 2015-2021 Altova GmbH

2200 The object is no longer valid.
2201 Invalid address for the return parameter was specified.

25.3.2.21.4 ContentOfNillableElementsIsNonMandatory

Property: ContentOfNillableElementsIsNonMandatory as Boolean

Description
If true, the contents of elements that are nillable will not be treated as mandatory.

Errors
2200 The object is no longer valid.
2201 Invalid address for the return parameter was specified.

25.3.2.21.5 FillAttributesWithSampleData

Property: FillAttributesWithSampleData as Boolean

Description
If true, attributes will have sample content.

Errors
2200 The object is no longer valid.
2201 Invalid address for the return parameter was specified.

25.3.2.21.6 FillElementsWithSampleData

Property: FillElementsWithSampleData as Boolean

Description
If true, elements will have sample content.

Errors
2200 The object is no longer valid.
2201 Invalid address for the return parameter was specified.

25.3.2.21.7 FillWithSampleData – obsolete

Property: FillWithSampleData as Boolean

Description
Do no longer access this property. Use FillAttributesWithSampleData 1275 and FillElementsWithSampleData 1275, instead.

Errors
0001 The property is no longer accessible.

25.3.2.21.8 LocalNameOfRootElement

Property: LocalNameOfRootElement as String

Description
Specifies the local name of the root element for the generated sample XML.

Errors
2200 The object is no longer valid.
2201 Invalid address for the return parameter was specified.

25.3.2.21.9 NamespaceURIOfRootElement

Property: NamespaceURIOfRootElement as String

Description
Specifies the namespace URI of the root element for the generated sample XML.

Errors
2200 The object is no longer valid.
2201 Invalid address for the return parameter was specified.

25.3.2.21.10 NonMandatoryAttributes

Property: NonMandatoryAttributes as Boolean

Description
If true attributes which are not mandatory are created in the sample XML instance file.

Errors
2200 The object is no longer valid.
2201 Invalid address for the return parameter was specified.

25.3.2.21.11 NonMandatoryElements

Property: NonMandatoryElements as Boolean

Description
If `true`, elements which are not mandatory are created in the sample XML instance file.

Errors

(C) 2015-2021 Altova GmbH

2200 The object is no longer valid.
2201 Invalid address was specified for the return parameter.

25.3.2.21.12 Optimization – obsolete

Property: Optimization as SPYSampleXMLGenerationOptimization 1366

Description
Do not use this property any longer. Use ChoiceMode and NonMandatoryElements.

Errors
0001 The property is no longer accessible.

25.3.2.21.13 OptionsDialogAction

Property: OptionsDialogAction as SPYDialogAction 1361

Description
To allow your script to fill in the default values and let the user see and react on the dialog, set this property to the value *spyDialogUserInput(2)*. If you want your script to define all the options in the schema documentation dialog without any user interaction necessary, use *spyDialogOK(0)*. Default is *spyDialogOK.*

Errors
2200 The object is no longer valid.
2201 Invalid value has been used to set the property.
Invalid address for the return parameter was specified.

25.3.2.21.14 Parent

Property: Parent as Dialogs 1215 (read-only)

Description
Access the parent of the object.

Errors
2200 The object is no longer valid.
2201 Invalid address for the return parameter was specified.

25.3.2.21.15 RepeatCount

Property: RepeatCount as long

Description
Number of elements to create for repeated types.

Errors
2200 The object is no longer valid.
2201 Invalid address for the return parameter was specified.

25.3.2.21.16 SampleValueHints

Property: `SampleValueHints` as `SPYSampleXMLGenerationSampleValueHints` 1367

Description
Specifies how to select data for the generated sample file.

Errors
2200 The object is no longer valid.
2201 Invalid address for the return parameter was specified.

25.3.2.21.17 SchemaOrDTDAssignment

Property: SchemaOrDTDAssignment as SPYSampleXMLGenerationSchemaOrDTDAssignment 1367

Description
Specifies in which way a reference to the related schema or DTD - which is this document - will be generated into the sample XML.

Errors
2200 The object is no longer valid.
2201 Invalid address for the return parameter was specified.

25.3.2.21.18 TakeFirstChoice – obsolete

Property: TakeFirstChoice as Boolean

Description
Do no longer use this property.

Errors
0001 The property is no longer accessible.

25.3.2.21.19 TryToUseNonAbstractTypes

Property: TryToUseNonAbstractTypes as Boolean

(C) 2015-2021 Altova GmbH

Description
If true, tries to use a non-abstract type for xsi:type, if element has an abstract type.

Errors
2200 The object is no longer valid.
2201 Invalid address for the return parameter was specified.

25.3.2.22 GridView

Methods
Deselect 1282
Select 1282

SetFocus 1282

Properties
CurrentFocus 1281

IsVisible 1282

Description
GridView Class

25.3.2.22.1 Events

25.3.2.22.1.1 OnBeforeDrag

Event: OnBeforeDrag() as Boolean

XMLSpy scripting environment - VBScript:
```
Function On_BeforeDrag()
        ' On_BeforeStartEditing = False  ' to prohibit dragging
End Function
```

XMLSpy scripting environment - JScript:
```
function On_BeforeDrag()
{
        // return false;  /* to prohibit dragging */
}
```

XMLSpy IDE Plugin:
IXMLSpyPlugIn.OnEvent (4, ...) // nEventId = 4

Description
This event gets fired on an attempt to drag an XMLData element on the grid view. Return *false* to prevent dragging the data element to a different position.

25.3.2.22.1.2 OnBeforeDrop

Event: OnBeforeDrop(*objXMLData* as XMLData[1309]) as Boolean

XMLSpy scripting environment - VBScript:

```
Function On_BeforeDrop(objXMLData)
        ' On_BeforeStartEditing = False  ' to prohibit dropping
End Function
```

XMLSpy scripting environment - JScript:

```
function On_BeforeDrop(objXMLData)
{
        // return false;  /* to prohibit dropping */
}
```

XMLSpy IDE Plugin:

IXMLSpyPlugIn.OnEvent (5, ...) // nEventId = 5

Description

This event gets fired on an attempt to drop a previously dragged XMLData element on the grid view. Return *false* to prevent the data element to be moved from its original position to the drop destination position.

25.3.2.22.1.3 OnBeforeStartEditing

Event: OnBeforeStartEditing(*objXMLData* as XMLData[1309], *bEditingName* as Boolean)as Boolean

XMLSpy scripting environment - VBScript:

```
Function On_BeforeStartEditing(objXMLData, bEditingName)
        ' On_BeforeStartEditing = False  ' to prohibit editing the field
End Function
```

XMLSpy scripting environment - JScript:

```
function On_BeforeStartEditing(objXMLData, bEditingName)
{
        // return false;  /* to prohibit editing the field */
}
```

XMLSpy IDE Plugin:

IXMLSpyPlugIn.OnEvent (1, ...) // nEventId = 1

Description

This event gets fired before the editing mode for a grid cell gets entered. If the parameter *bEditingName* is true, the name part of the element will be edited, it its value is false, the value part will be edited.

(C) 2015-2021 Altova GmbH

25.3.2.22.1.4 OnEditingFinished

Event: OnEditingFinished(*objXMLData* as XMLData 1309, *bEditingName* as Boolean)

XMLSpy scripting environment - VBScript:
Function On_EditingFinished(*objXMLData*, *bEditingName*)
End Function

XMLSpy scripting environment - JScript:
function On_EditingFinished(*objXMLData*, *bEditingName*)
{
}

XMLSpy IDE Plugin:
IXMLSpyPlugIn.OnEvent (2, ...) // nEventId = 2

Description
This event gets fired when the editing mode of a grid cell is exited. The parameter *bEditingName* specifies if the name part of the element has been edited.

25.3.2.22.1.5 OnFocusChanged

Event: OnFocusChanged(*objXMLData* as XMLData 1309, *bSetFocus* as Boolean, *bEditingName* as Boolean)

XMLSpy scripting environment - VBScript:
Function On_FocusChanged(*objXMLData*, *bSetFocus*, *bEditingName*)
End Function

XMLSpy scripting environment - JScript:
function On_FocusChanged(*objXMLData*, *bSetFocus*, *bEditingName*)
{
}

XMLSpy IDE Plugin:
IXMLSpyPlugIn.OnEvent (3, ...) // nEventId = 3

Description
This event gets fired whenever a grid cell receives or loses the cursor focus. If the parameter *bEditingName* is *true*, focus of the name part of the grid element has changed. Otherwise, focus of the value part has changed.

25.3.2.22.2 CurrentFocus

Property: CurrentFocus as XMLData 1309

Description
Holds the XML element with the current focus. This property is read-only.

25.3.2.22.3 Deselect

Method: Deselect(*pData* as XMLData[1309])

Description
Deselects the element pData in the grid view.

25.3.2.22.4 IsVisible

Property: IsVisible as Boolean

Description
True if the grid view is the active view of the document. This property is read-only.

25.3.2.22.5 Select

Method: Select (*pData* as XMLData[1309])

Description
Selects the XML element pData in the grid view.

25.3.2.22.6 SetFocus

Method: SetFocus (*pFocusData* as XMLData[1309])

Description
Sets the focus to the element pFocusData in the grid view.

25.3.2.23 SchemaDocumentationDlg

Properties and Methods

Standard automation properties
Application[1284]
Parent[1290]

Interaction and visibility properties
OutputFile[1290]
OutputFileDialogAction[1290]
OptionsDialogAction[1289]
ShowProgressBar[1293]
ShowResult[1294]

Document generation options and methods
OutputFormat[1290]
UseFixedDesign[1296]
SPSFile[1295]

(C) 2015-2021 Altova GmbH

EmbedDiagrams 1285
DiagramFormat 1284
MultipleOutputFiles 1289
EmbedCSSInHTML 1285
CreateDiagramsFolder 1284
GenerateRelativeLinks 1285

IncludeAll 1285
IncludeIndex 1287
IncludeGlobalAttributes 1286
IncludeGlobalElements 1287
IncludeLocalAttributes 1287
IncludeLocalElements 1288
IncludeGroups 1287
IncludeComplexTypes 1286
IncludeSimpleTypes 1289
IncludeAttributeGroups 1286
IncludeRedefines 1288
IncludeReferencedSchemas 1288

AllDetails 1283
ShowDiagram 1292
ShowNamespace 1293
ShowType 1295
ShowChildren 1291
ShowUsedBy 1295
ShowProperties 1293
ShowSingleFacets 1294
ShowPatterns 1293
ShowEnumerations 1292
ShowAttributes 1291
ShowIdentityConstraints 1292
ShowAnnotations 1291
ShowSourceCode 1294

Description
This object combines all options for schema document generation as they are available through user interface dialog boxes in XMLSpy. The document generation options are initialized with the values used during the last generation of schema documentation. However, before using the object you have to set the SetOutputFile 1290 property to a valid file path. Use OptionsDialogAction 1289, OutputFileDialogAction 1290 and ShowProgressBar 1293 to specify the level of user interaction desired. You can use IncludeAll 1285 and AllDetails 1283 to set whole option groups at once or the individual properties to operate on a finer granularity.

25.3.2.23.1 AllDetails

Method: AllDetails (i_bDetailsOn as Boolean)

Description
Use this method to turn all details options on or off.

Errors
2900 The object is no longer valid.

25.3.2.23.2 Application

Property: Application as Application[1131] (read-only)

Description
Access the XMLSpy application object.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

25.3.2.23.3 CreateDiagramsFolder

Property: CreateDiagramsFolder as Boolean

Description
Set this property to true, to create a directory for the created images. Otherwise the diagrams will be created next to the documentation. This property is only available when the diagrams are not embedded. The default for the first run is false.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

25.3.2.23.4 DiagramFormat

Property: DiagramFormat as SPYImageKind[1363]

Description
This property specifies the generated diagram image type. This property is not available for HTML documentation. The property is initialized with the value used during the last call to Document.GenerateSchemaDocumentation[1234]. The default for the first run is PNG.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

(C) 2015-2021 Altova GmbH

25.3.2.23.5 EmbedCSSInHTML

Property: EmbedCSSInHTML as Boolean

Description
Set this property to true, to embed the CSS data in the generated HTML document. Otherwise a separate file will be created and linked. This property is only available for HTML documentation. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

25.3.2.23.6 EmbedDiagrams

Property: EmbedDiagrams as Boolean

Description
Set this property to true, to embed the diagrams in the generated document. This property is not available for HTML documentation. The property is initialized with the value used during the last call to Document.GenerateSchemaDocumentation 1234. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

25.3.2.23.7 GenerateRelativeLinks

Property: GenerateRelativeLinks as Boolean

Description
Set this property to true, to create relative paths to local files. This property is not available for HTML documentation. The property is initialized with the value used during the last call to Document.GenerateSchemaDocumentation 1234. The default for the first run is false.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

25.3.2.23.8 IncludeAll

Method: IncludeAll (i_bInclude as Boolean)

Description
Use this method to mark or unmark all include options.

Errors
2900 The object is no longer valid.

25.3.2.23.9 IncludeAttributeGroups

Property: IncludeAttributeGroups as Boolean

Description
Set this property to true, to include attribute groups in the schema documentation. The property is initialized with the value used during the last call to Document.GenerateSchemaDocumentation[1234]. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

25.3.2.23.10 IncludeComplexTypes

Property: IncludeComplexTypes as Boolean

Description
Set this property to true, to include complex types in the schema documentation. The property is initialized with the value used during the last call to Document.GenerateSchemaDocumentation[1234]. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

25.3.2.23.11 IncludeGlobalAttributes

Property: IncludeGlobalAttributes as Boolean

Description
Set this property to true, to include global attributes in the schema documentation. The property is initialized with the value used during the last call to Document.GenerateSchemaDocumentation[1234]. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

(C) 2015-2021 Altova GmbH

25.3.2.23.12 IncludeGlobalElements

Property: IncludeGlobalElements as Boolean

Description
Set this property to true, to include global elements in the schema documentation. The property is initialized with the value used during the last call to Document.GenerateSchemaDocumentation[1234]. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

25.3.2.23.13 IncludeGroups

Property: IncludeGroups as Boolean

Description
Set this property to true, to include groups in the schema documentation. The property is initialized with the value used during the last call to Document.GenerateSchemaDocumentation[1234]. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

25.3.2.23.14 IncludeIndex

Property: IncludeIndex as Boolean

Description
Set this property to true, to include an index in the schema documentation. The property is initialized with the value used during the last call to Document.GenerateSchemaDocumentation[1234]. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

25.3.2.23.15 IncludeLocalAttributes

Property: IncludeLocalAttributes as Boolean

Description

Set this property to true, to include local attributes in the schema documentation. The property is initialized with the value used during the last call to Document.GenerateSchemaDocumentation[1234]. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

25.3.2.23.16 IncludeLocalElements

Property: IncludeLocalElements as Boolean

Description
Set this property to true, to include local elements in the schema documentation. The property is initialized with the value used during the last call to Document.GenerateSchemaDocumentation[1234]. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

25.3.2.23.17 IncludeRedefines

Property: IncludeRedefines as Boolean

Description
Set this property to true, to include redefines in the schema documentation. The property is initialized with the value used during the last call to Document.GenerateSchemaDocumentation[1234]. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

25.3.2.23.18 IncludeReferencedSchemas

Property: IncludeReferencedSchemas as Boolean

Description
Set this property to true, to include referenced schemas in the schema documentation. The property is initialized with the value used during the last call to Document.GenerateSchemaDocumentation[1234]. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

(C) 2015-2021 Altova GmbH

25.3.2.23.19 IncludeSimpleTypes

Property: IncludeSimpleTypes as Boolean

Description
Set this property to true, to include simple types in the schema documentation. The property is initialized with the value used during the last call to Document.GenerateSchemaDocumentation[1234]. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

25.3.2.23.20 MultipleOutputFiles

Property: MultipleOutputFiles as Boolean

Description
Set this property to true, to split the documentation files. The property is initialized with the value used during the last call to Document.GenerateSchemaDocumentation[1234]. The default for the first run is false.

Errors
2900 The object is no longer valid.
2901 Invalid value has been used to set the property.
Invalid address for the return parameter was specified.

25.3.2.23.21 OptionsDialogAction

Property: OptionsDialogAction as SPYDialogAction[1361]

Description
To allow your script to fill in the default values and let the user see and react on the dialog, set this property to the value *spyDialogUserInput(2)*. If you want your script to define all the options in the schema documentation dialog without any user interaction necessary, use *spyDialogOK(0)*. Default is *spyDialogOK.*

Errors
2900 The object is no longer valid.
2901 Invalid value has been used to set the property.
Invalid address for the return parameter was specified.

25.3.2.23.22 OutputFile

Property: OutputFile as String

Description
Full path and name of the file that will contain the generated documentation. In case of HTML output, additional '.png' files will be generated based on this filename. The default value for this property is an empty string and needs to be replaced before using this object in a call to Document.GenerateSchemaDocumentation[1234].

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

25.3.2.23.23 OutputFileDialogAction

Property: OutputFileDialogAction as SPYDialogAction[1361]

Description
To allow the user to select the output file with a file selection dialog, set this property to *spyDialogUserInput(2)*. If the value stored in OutputFile[1290] should be taken and no user interaction should occur, use *spyDialogOK(0)*. Default is *spyDialogOK.*

Errors
2900 The object is no longer valid.
2901 Invalid value has been used to set the property.
Invalid address for the return parameter was specified.

25.3.2.23.24 OutputFormat

Property: OutputFormat as SPYSchemaDocumentationFormat[1368]

Description
Defines the kind of documentation that will be generated: HTML (value=`0`), MS-Word (value=`1`), or RTF (value=`2`). The property gets initialized with the value used during the last call to Document.GenerateSchemaDocumentation[1234]. The default for the first run is HTML.

Errors
2900 The object is no longer valid.
2901 Invalid value has been used to set the property.
Invalid address for the return parameter was specified.

25.3.2.23.25 Parent

Property: Parent as Dialogs[1215] (read-only)

(C) 2015-2021 Altova GmbH

Description
Access the parent of the object.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

25.3.2.23.26 ShowAnnotations

Property: ShowAnnotations as Boolean

Description
Set this property to true, to show the annotations to a type definition in the schema documentation. The property is initialized with the value used during the last call to Document.GenerateSchemaDocumentation 1234. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

25.3.2.23.27 ShowAttributes

Property: ShowAttributes as Boolean

Description
Set this property to true, to show the type definitions attributes in the schema documentation. The property is initialized with the value used during the last call to Document.GenerateSchemaDocumentation 1234. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

25.3.2.23.28 ShowChildren

Property: ShowChildren as Boolean

Description
Set this property to true, to show the children of a type definition as links in the schema documentation. The property is initialized with the value used during the last call to Document.GenerateSchemaDocumentation 1234. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

25.3.2.23.29 ShowDiagram

Property: ShowDiagram as Boolean

Description
Set this property to true, to show type definitions as diagrams in the schema documentation. The property is initialized with the value used during the last call to Document.GenerateSchemaDocumentation[1234]. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

25.3.2.23.30 ShowEnumerations

Property: ShowEnumerations as Boolean

Description
Set this property to true, to show the enumerations contained in a type definition in the schema documentation. The property is initialized with the value used during the last call to Document.GenerateSchemaDocumentation[1234]. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

25.3.2.23.31 ShowIdentityConstraints

Property: ShowIdentityConstraints as Boolean

Description
Set this property to true, to show a type definitions identity constraints in the schema documentation. The property is initialized with the value used during the last call to Document.GenerateSchemaDocumentation[1234]. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

(C) 2015-2021 Altova GmbH

25.3.2.23.32 ShowNamespace

Property: ShowNamespace as Boolean

Description
Set this property to true, to show the namespace of type definitions in the schema documentation. The property is initialized with the value used during the last call to Document.GenerateSchemaDocumentation 1234. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

25.3.2.23.33 ShowPatterns

Property: ShowPatterns as Boolean

Description
Set this property to true, to show the patterns of a type definition in the schema documentation. The property is initialized with the value used during the last call to Document.GenerateSchemaDocumentation 1234. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

25.3.2.23.34 ShowProgressBar

Property: ShowProgressBar as Boolean

Description
Set this property to true, to make the window showing the document generation progress visible. Use false, to hide it. Default is false.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

25.3.2.23.35 ShowProperties

Property: ShowProperties as Boolean

Description

Set this property to true, to show the type definition properties in the schema documentation. The property is initialized with the value used during the last call to Document.GenerateSchemaDocumentation[1234]. The default for the first run is true.

Errors

2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

25.3.2.23.36 ShowResult

Property: ShowResult as Boolean

Description

Set this property to true, to automatically open the resulting document when generation was successful. HTML documentation will be opened in XMLSpy. To show Word documentation, MS-Word will be started. The property gets initialized with the value used during the last call to Document.GenerateSchemaDocumentation[1234]. The default for the first run is true.

Errors

2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

25.3.2.23.37 ShowSingleFacets

Property: ShowSingleFacets as Boolean

Description

Set this property to true, to show the facets of a type definition in the schema documentation. The property is initialized with the value used during the last call to Document.GenerateSchemaDocumentation[1234]. The default for the first run is true.

Errors

2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

25.3.2.23.38 ShowSourceCode

Property: ShowSourceCode as Boolean

Description

Set this property to true, to show the XML source code for type definitions in the schema documentation. The property is initialized with the value used during the last call to Document.GenerateSchemaDocumentation[1234]. The default for the first run is true.

Errors

(C) 2015-2021 Altova GmbH

2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

25.3.2.23.39 ShowType

Property: ShowType as Boolean

Description
Set this property to true, to show the type of type definitions in the schema documentation. The property is initialized with the value used during the last call to Document.GenerateSchemaDocumentation 1234. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

25.3.2.23.40 ShowUsedBy

Property: ShowUsedBy as Boolean

Description
Set this property to true, to show the used-by relation for type definitions in the schema documentation. The property is initialized with the value used during the last call to Document.GenerateSchemaDocumentation 1234. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

25.3.2.23.41 SPSFile

Property: SPSFile as String

Description
Full path and name of the SPS file that will be used to generate the documentation.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

25.3.2.23.42 UseFixedDesign

Property: UseFixedDesign as Boolean

Description
Specifies whether the documentation should be created with a fixed design or with a design specified by a SPS file (which requires StyleVision).

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

25.3.2.24 SpyProject

Methods
CloseProject 1296
SaveProject 1297
SaveProjectAs 1297

Properties
RootItems 1297
ProjectFile 1296

Description
SpyProject Class

25.3.2.24.1 CloseProject

Declaration: CloseProject(*bDiscardChanges* as Boolean, *bCloseFiles* as Boolean, *bDialog* as Boolean)

Parameters
bDiscardChanges
Set bDiscardChanges to FALSE if you want to save the changes of the open project files and the project.

bCloseFiles
Set bCloseFiles to TRUE to close all open project files.

bDialog
Show dialogs for user input.

Description
CloseProject closes the current project.

25.3.2.24.2 ProjectFile

Declaration: ProjectFile as String

(C) 2015-2021 Altova GmbH

Description
Path and filename of the project.

25.3.2.24.3 RootItems

Declaration: RootItems as SpyProjectItems 1300

Description
Root level of collection of project items.

25.3.2.24.4 SaveProject

Declaration: SaveProject

Description
SaveProject saves the current project.

25.3.2.24.5 SaveProjectAs

Declaration: SaveProjectAs (strPath as String, bDialog as Boolean)

Parameters
strPath
Full path with file name of new project file.

bDialog
If bDialog is TRUE, a file-dialog will be displayed.

Description
SaveProjectAs stores the project data into a new location.

25.3.2.25 SpyProjectItem

Methods
Open 1298

Properties
ChildItems 1298
ParentItem 1298
FileExtensions 1298
ItemType 1298
Name 1298
Path 1299
ValidateWith 1299
XMLForXSLTransformation 1299
XSLForXMLTransformation 1299
XSLTransformationFileExtension 1299
XSLTransformationFolder 1299

Description
SpyProjectItem Class

25.3.2.25.1 ChildItems

Declaration: ChildItems as SpyProjectItems 1300

Description
If the item is a folder, ChildItems is the collection of the folder content.

25.3.2.25.2 FileExtensions

Declaration: FileExtensions as String

Description
Used to set the file extensions if the project item is a folder.

25.3.2.25.3 ItemType

Declaration: ItemType as SPYProjectItemTypes 1365

Description
This property is read-only.

25.3.2.25.4 Name

Declaration: Name as String

Description
Name of the project item. This property is read-only.

25.3.2.25.5 Open

Declaration: Open as Document 1218

Return Value
The project item opened as document.

Description
Opens the project item.

25.3.2.25.6 ParentItem

Declaration: ParentItem as SpyProjectItem 1297

(C) 2015-2021 Altova GmbH

Description
Parent item of the current project item. Can be NULL (Nothing) if the project item is a top-level item.

25.3.2.25.7 Path

Declaration: Path as String

Description
Path of project item. This property is read-only.

25.3.2.25.8 ValidateWith

Declaration: ValidateWith as String

Description
Used to set the schema/DTD for validation.

25.3.2.25.9 XMLForXSLTransformation

Declaration: XMLForXSLTransformation as String

Description
Used to set the XML for XSL transformation.

25.3.2.25.10 XSLForXMLTransformation

Declaration: XSLForXMLTransformation as String

Description
Used to set the XSL for XML transformation.

25.3.2.25.11 XSLTransformationFileExtension

Declaration: XSLTransformationFileExtension as String

Description
Used to set the file extension for XSL transformation output files.

25.3.2.25.12 XSLTransformationFolder

Declaration: XSLTransformationFolder as String

Description
Used to set the destination folder for XSL transformation output files.

25.3.2.26 SpyProjectItems

Methods
AddFile 1300
AddFolder 1300
AddURL 1300
RemoveItem 1301

Properties
Count 1301
Item 1301

Description
SpyProjectItems Class

25.3.2.26.1 AddFile

Declaration: AddFile (*strPath* as String)

Parameters

strPath
Full path with file name of new project item

Description
The method adds a new file to the collection of project items.

25.3.2.26.2 AddFolder

Declaration: AddFolder (*strName* as String)

Parameters

strName
Name of the new folder.

Description
The method AddFolder adds a folder with the name strName to the collection of project items.

25.3.2.26.3 AddURL

Declaration: AddURL (*strURL* as String, *nURLType* as SPYURLTypes 1369, *strUser* as String, *strPassword* as String, *bSave* as Boolean)

Description

strURL
URL to open as document.

(C) 2015-2021 Altova GmbH

nURLType
Type of document to open. Set to -1 for auto detection.

strUser
Name of the user if required. Can be empty.

strPassword
Password for authentification. Can be empty.

bSave
Save user and password information.

Description
The method adds an URL item to the project collection.

25.3.2.26.4 Count

Declaration: Count as long

Description
This property gets the count of project items in the collection. The property is read-only.

25.3.2.26.5 Item

Declaration: Item (*n* as long) as SpyProjectItem 1297

Description
Retrieves the n-th element of the collection of project items. The first item has index 1.

25.3.2.26.6 RemoveItem

Declaration: RemoveItem (*pItem* as SpyProjectItem 1297)

Description
RemoveItem deletes the item pItem from the collection of project items.

25.3.2.27 TextImportExportSettings

Properties for import only
ImportFile 1303

Properties for export only
DestinationFolder 1302
FileExtension 1303
CommentIncluded 1302
RemoveDelimiter 1303
RemoveNewline 1303

Properties for import and export

HeaderRow 1303
FieldDelimiter 1303
EnclosingCharacter 1302
Encoding 1302
EncodingByteOrder 1302

Description
TextImportExportSettings contains options common to text import and export functions.

25.3.2.27.1 CommentIncluded

Property: CommentIncluded as Boolean

Description
This property tells whether additional comments are added to the generated text file. Default is true. This property is used only when exporting to text files.

25.3.2.27.2 DestinationFolder

Property: DestinationFolder as String

Description
The property DestinationFolder sets the folder where the created files are saved during text export.

25.3.2.27.3 EnclosingCharacter

Property: EnclosingCharacter as SPYTextEnclosing 1369

Description
This property defines the character that encloses all field values for import and export. Default is spyNoEnclosing 1369.

25.3.2.27.4 Encoding

Property: Encoding as String

Description
The property Encoding sets the character encoding for the text files for importing and exporting.

25.3.2.27.5 EncodingByteOrder

Property: EncodingByteOrder as SPYEncodingByteOrder 1362

Description
The property EncodingByteOrder sets the byte order for Unicode characters. Default is spyNONE 1362.

(C) 2015-2021 Altova GmbH

25.3.2.27.6 FieldDelimiter

Property: FieldDelimiter as SPYTextDelimiters 1369

Description
The property FieldDelimiter defines the delimiter between the fields during import and export. Default is spyTabulator 1369.

25.3.2.27.7 FileExtension

Property: FileExtension as String

Description
This property sets the file extension for files created on text export.

25.3.2.27.8 HeaderRow

Property: HeaderRow as Boolean

Description
The property HeaderRow is used during import and export. Set HeaderRow true on import, if the first line of the text file contains the names of the columns. Set HeaderRow true on export, if the first line in the created text files should contain the name of the columns. Default value is true.

25.3.2.27.9 ImportFile

Property: ImportFile as String

Description
This property is used to set the text file for import. The string has to be a full qualified path. See also Import and Export 1103.

25.3.2.27.10 RemoveDelimiter

Property: RemoveDelimiter as Boolean

Description
The property RemoveDelimiter defines whether characters in the text that are equal to the delimiter character are removed. Default is false. This property is used only when exporting to text files.

25.3.2.27.11 RemoveNewline

Property: RemoveNewline as Boolean

Description

The property RemoveNewline defines whether newline characters in the text are removed. Default is false. This property is used only when exporting to text files.

25.3.2.28 TextView

Properties and Methods

Application 1305
Parent 1307

LineFromPosition 1307
PositionFromLine 1308
LineLength 1307
SelText 1309
GetRangeText 1306
ReplaceText 1308
MoveCaret 1307
GoToLineChar 1306
SelectText 1309
SelectionStart 1308
SelectionEnd 1308
Text 1309
LineCount 1306
Length 1306

Description

25.3.2.28.1 Events

25.3.2.28.1.1 OnBeforeShowSuggestions

Event: OnBeforeShowSuggestions() as Boolean

Description
This event gets fired before a suggestion window is shown. The Document 1218 property Suggestions 1244 contains a string array that is recommended to the user. It is possible to modify the displayed recommendations during this event. Before doing so you have to assign an empty array to the Suggestions 1244 property. The best location for this is the OnDocumentOpened 1133 event. To prevent the suggestion window to show up return false and true to continue its display.

Examples
Given below are examples of how this event can be scripted.

XMLSpy scripting environment - VBScript:
```
Function On_BeforeShowSuggestions()
End Function
```

XMLSpy scripting environment - JScript:
```
function On_BeforeShowSuggestions()
{
```

}

XMLSpy IDE Plugin:
IXMLSpyPlugIn.OnEvent (33, ...)// nEventId = 33

25.3.2.28.1.2 OnChar

Event: OnChar(nChar as Long, bExistSuggestion as Boolean) as Boolean

Description
This event gets fired on each key stroke. The parameter nChar is the key that was pressed and bExistSuggestions tells whether a XMLSpy generated suggestions window is displayed after this key. The Document[1218] property Suggestions[1244] contains a string array that is recommended to the user. It is possible to modify the displayed recommendations during this event. Before doing so you have to assign an empty array to the Suggestions[1244] property. The best location for this is the OnDocumentOpened[1133] event. To prevent the suggestion window to show up return false and true to continue its display.
It is also possible to create a new suggestions window when none is provided by XMLSpy. Set the Document[1218] property Suggestions[1244] to a string array with your recommendations and return true.
This event is fired before the OnBeforeShowSuggestions[1304] event. If you prevent to show the suggestion window by returning false then OnBeforeShowSuggestions[1304] is not fired.

Examples
Given below are examples of how this event can be scripted.

XMLSpy scripting environment - VBScript:
```
Function On_Char(nChar, bExistSuggestions)
End Function
```

XMLSpy scripting environment - JScript:
```
function On_Char(nChar, bExistSuggestions)
{
}
```

XMLSpy IDE Plugin:
IXMLSpyPlugIn.OnEvent (35, ...)// nEventId = 35

25.3.2.28.2 Application

Property: Application as Application[1131] (read-only)

Description
Access the XMLSpy application object.

Errors

3900	The object is no longer valid.
3901	Invalid address for the return parameter was specified.

25.3.2.28.3 GetRangeText

Method: GetRangeText(nStart as Long, nEnd as Long) as String

Description
Returns the text in the specified range.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

25.3.2.28.4 GoToLineChar

Method: GoToLineChar(nLine as Long, nChar as Long)

Description
Moves the caret to the specified line and character position.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

25.3.2.28.5 Length

Property: Length as Long

Description
Returns the character count of the document.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

25.3.2.28.6 LineCount

Property: LineCount as Long

Description
Returns the number of lines in the document.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

(C) 2015-2021 Altova GmbH

25.3.2.28.7 LineFromPosition

Method: LineFromPosition(nCharPos as Long) as Long

Description
Returns the line number of the character position.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

25.3.2.28.8 LineLength

Method: LineLength(nLine as Long) as Long

Description
Returns the length of the line.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

25.3.2.28.9 MoveCaret

Method: MoveCaret(nDiff as Long)

Description
Moves the caret nDiff characters.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

25.3.2.28.10 Parent

Property: Parent as Document 1218 (read-only)

Description
Access the parent of the object.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

25.3.2.28.11 PositionFromLine

Method: PositionFromLine(nLine as Long) as Long

Description
Returns the start position of the line.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

25.3.2.28.12 ReplaceText

Method: ReplaceText(nPosFrom as Long, nPosTill as Long, sText as String)

Description
Replaces the text in the specified range.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

25.3.2.28.13 SelectionEnd

Property: SelectionEnd as Long

Description
Returns/sets the text selection end position.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

25.3.2.28.14 SelectionStart

Property: SelectionStart as Long

Description
Returns/sets the text selection start position.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

(C) 2015-2021 Altova GmbH

25.3.2.28.15 SelectText

Method: SelectText(nPosFrom as Long, nPosTill as Long)

Description
Selects the text in the specified range.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

25.3.2.28.16 SelText

Property: SelText as String

Description
Returns/sets the selected text.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

25.3.2.28.17 Text

Property: Text as String

Description
Returns/sets the document text.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

25.3.2.29 XMLData

Properties
Kind 1319
Name 1320
TextValue 1320

HasChildren 1317
MayHaveChildren 1319
Parent 1320

Methods
GetFirstChild 1315

GetNextChild 1316
GetCurrentChild 1315

InsertChild 1318
InsertChildAfter 1318
InsertChildBefore 1318
AppendChild 1311

EraseAllChildren 1312
EraseChild 1312
EraseCurrentChild 1312

IsSameNode 1319

CountChildren 1311
CountChildrenKind 1311

GetChild 1313
GetChildAttribute 1314
GetChildElement 1314
GetChildKind 1314
GetNamespacePrefixForURI 1315

HasChildrenKind 1317
SetTextValueXMLEncoded 1320

Description
The XMLData interface provides direct XML-level access to a document. You can read and directly modify the XML representation of the document. However, please, note the following restrictions:

- The XMLData representation is only valid when the document is shown in grid view or authentic view.
- When in authentic view, additional XMLData elements are automatically inserted as parents of each visible document element. Typically this is an XMLData of kind spyXMLDataElement with the Name 1320 property set to 'Text'.
- When you use the XMLData interface while in a different view mode you will not receive errors, but changes are not reflected to the view and might get lost during the next view switch.

Note also:

- Setting a new text value for an XML element is possible if the element does not have non-text children. A text value can be set even if the element has attributes.
- When setting a new text value for an XML element which has more than one text child, the latter will be deleted and replaced by one new text child.
- When reading the text value of an XML element which has more than one text child, only the value of the first text child will be returned.

Objects of this class represent the different atomic parts of an XML document. See the enumeration type SPYXMLDataKind 1371 for the available part types. Each part knows its children, thus forming a XMLData tree with Document.RootElement 1240 at its top. To get the top element of the document content - ignoring the XML header - use Document.DataRoot 1228. For an examples on how to traverse the XMLData tree, see GetNextChild 1316.

25.3.2.29.1 AppendChild

Declaration: AppendChild (*pNewData* as XMLData[1309])

Description
AppendChild appends pNewData as last child to the XMLData object.

Errors
1500 The XMLData object is no longer valid.
1505 Invalid XMLData kind was specified.
1506 Invalid address for the return parameter was specified.
1507 Element cannot have Children
1512 Cyclic insertion - new data element is already part of document
1514 Invalid XMLData kind was specified for this position.
1900 Document must not be modified

Example

```
Dim objCurrentParent As XMLData
Dim objNewChild As XMLData

Set objNewChild = objSpy.ActiveDocument.CreateChild(spyXMLDataElement)
Set objCurrentParent = objSpy.ActiveDocument.RootElement

objCurrentParent.AppendChild objNewChild

Set objNewChild = Nothing
```

25.3.2.29.2 CountChildren

Declaration: CountChildren as long

Description
CountChildren gets the number of children.

Available with TypeLibrary version 1.5

Errors
1500 The XMLData object is no longer valid.

25.3.2.29.3 CountChildrenKind

Declaration: CountChildrenKind (*nKind* as SPYXMLDataKind[1371]) as long

Description
CountChildrenKind gets the number of children of the specific kind.

Available with TypeLibrary version 1.5

Errors
1500 The XMLData object is no longer valid.

25.3.2.29.4 EraseAllChildren

Declaration: EraseAllChildren

Description
EraseAllChildren deletes all associated children of the XMLData object.

Errors
1500 The XMLData object is no longer valid.
1900 Document must not be modified

Example
The sample erases all elements of the active document.

```
Dim objCurrentParent As XMLData

Set objCurrentParent = objSpy.ActiveDocument.RootElement
objCurrentParent.EraseAllChildren
```

25.3.2.29.5 EraseChild

Method: EraseChild (Child as XMLData[1309])

Description
Deletes the given child node.

Errors
1500 Invalid object.
1506 Invalid input xml
1510 Invalid parameter.

25.3.2.29.6 EraseCurrentChild

Declaration: EraseCurrentChild

Description
EraseCurrentChild deletes the current XMLData child object. Before you call EraseCurrentChild you must initialize an internal iterator with XMLData.GetFirstChild[1315]. After deleting the current child, EraseCurrentChild increments the internal iterator of the XMLData element. No error is returned when the last child gets erased

(C) 2015-2021 Altova GmbH

and the iterator is moved past the end of the child list. The next call to EraseCurrentChild however, will return error 1503.

Errors

1500 The XMLData object is no longer valid.
1503 No iterator is initialized for this XMLData object, or the iterator points past the last child.
1900 Document must not be modified

Examples

```
// ---------------------------------------
// XMLSpy scripting environment - JScript
// erase all children of XMLData
// ---------------------------------------
// let's get an XMLData element, we assume that the
// cursor selects the parent of a list in grid view
var objList = Application.ActiveDocument.GridView.CurrentFocus;

// the following line would be shorter, of course
//        objList.EraseAllChildren ();

// but we want to demonstrate the usage of EraseCurrentChild
if ((objList != null) && (objList.HasChildren))
{
        try
        {
                objEle = objList.GetFirstChild(-1);
                while (objEle != null)
                        objList.EraseCurrentChild();
                        // no need to call GetNextChild
        }
        catch (err)
                // 1503 - we reached end of child list
                { if ((err.number & 0xffff) != 1503) throw (err); }
}
```

25.3.2.29.7 GetChild

Declaration: GetChild (*position* as long) as XMLData 1309

Return Value

Returns an XML element as XMLData object.

Description

GetChild() returns a reference to the child at the given index (zero-based).

Available with TypeLibrary version 1.5

Errors

1500 The XMLData object is no longer valid.
1510 Invalid address for the return parameter was specified.

25.3.2.29.8 GetChildAttribute

Method: GetChildAttribute (strName as string) child as XMLData object (`NULL` on error)

Description
Retrieves the attribute having the given name.

Errors
1500 Invalid object.
1510 Invalid parameter.

25.3.2.29.9 GetChildElement

Method: GetChildElement (strName as string, nIndex as long) child as XMLData object (`NULL` on error)

Description
Retrieves the Nth child element with the given name.

Errors
1500 Invalid object.
1510 Invalid parameter.

25.3.2.29.10 GetChildKind

Declaration: GetChildKind (*position* as long, *nKind* as SPYXMLDataKind[1371]) as XMLData[1309]

Return Value
Returns an XML element as XMLData object.

Description
GetChildKind() returns a reference to a child of this kind at the given index (zero-based). The position parameter is relative to the number of children of the specified kind and not to all children of the object.

Available with TypeLibrary version 1.5

Errors
1500 The XMLData object is no longer valid.
1510 Invalid address for the return parameter was specified.

(C) 2015-2021 Altova GmbH

25.3.2.29.11 GetCurrentChild

Declaration: GetCurrentChild as XMLData 1309

Return Value
Returns an XML element as XMLData object.

Description
GetCurrentChild gets the current child. Before you call GetCurrentChild you must initialize an internal iterator with XMLData.GetFirstChild 1315.

Errors
1500 The XMLData object is no longer valid.
1503 No iterator is initialized for this XMLData object.
1510 Invalid address for the return parameter was specified.

25.3.2.29.12 GetFirstChild

Declaration: GetFirstChild (*nKind* as SPYXMLDataKind 1371) as XMLData 1309

Return Value
Returns an XML element as XMLData object.

Description
GetFirstChild initializes a new iterator and returns the first child. Set nKind = -1 to get an iterator for all kinds of children.
REMARK: The iterator is stored inside the XMLData object and gets destroyed when the XMLData object gets destroyed. Be sure to keep a reference to this object as long as you want to use GetCurrentChild 1315, GetNextChild 1316 or EraseCurrentChild 1312.

Errors
1500 The XMLData object is no longer valid.
1501 Invalid XMLData kind was specified.
1504 Element has no children of specified kind.
1510 Invalid address for the return parameter was specified.

Example
See the example at XMLData.GetNextChild 1316.

25.3.2.29.13 GetNamespacePrefixForURI

Method: GetNamespacePrefixForURI (strURI as string) strNS as string

Description
Returns the namespace prefix of the supplied URI.

Errors
1500 Invalid object.
1510 Invalid parameter.

25.3.2.29.14 GetNextChild

Declaration: GetNextChild as XMLData 1309

Return Value
Returns an XML element as XMLData object.

Description
GetNextChild steps to the next child of this element. Before you call GetNextChild you must initialize an internal iterator with XMLData.GetFirstChild 1315.

Check for the last child of the element as shown in the sample below.

Errors
1500 The XMLData object is no longer valid.
1503 No iterator is initialized for this XMLData object.
1510 Invalid address for the return parameter was specified.

Examples

```
' ---------------------------------------------
' VBA code snippet - iterate XMLData children
' ---------------------------------------------
On Error Resume Next
Set objParent = objSpy.ActiveDocument.RootElement

'get elements of all kinds
Set objCurrentChild = objParent.GetFirstChild(-1)

Do
        'do something useful with the child

        'step to next child
        Set objCurrentChild = objParent.GetNextChild
Loop Until (Err.Number - vbObjectError = 1503)
```

```
// ---------------------------------------
// XMLSpy scripting environment - JScript
// iterate through children of XMLData
// ---------------------------------------
try
{
        var objXMLData = ... // initialize somehow
        var objChild = objXMLData.GetFirstChild(-1);

        while (true)
```

```
        {
                // do something usefull with objChild

                objChild = objXMLData.GetNextChild();
        }
}
catch (err)
{
        if ((err.number & 0xffff) == 1504)
                ; // element has no children
        else if ((err.number & 0xffff) == 1503)
                ;       // last child reached
        else
                throw (err);
}
```

25.3.2.29.15 GetTextValueXMLDecoded

Method: GetTextValueXMLDecoded ()as string

Description
Gets the decoded text value of the XML.

Errors
1500 Invalid object.
1510 Invalid parameter.

25.3.2.29.16 HasChildren

Declaration: HasChildren as Boolean

Description
The property is true if the object is the parent of other XMLData objects. This property is read-only.

Errors
1500 The XMLData object is no longer valid.
1510 Invalid address for the return parameter was specified.

25.3.2.29.17 HasChildrenKind

Declaration: HasChildrenKind (*nKind* as SPYXMLDataKind 1371) as Boolean

Description
The method returns true if the object is the parent of other XMLData objects of the specific kind.

Available with TypeLibrary version 1.5

Errors
1500 The XMLData object is no longer valid.
1510 Invalid address for the return parameter was specified.

25.3.2.29.18 InsertChild

Declaration: InsertChild (*pNewData* as XMLData[1309])

Description
InsertChild inserts the new child before the current child (see also XMLData.GetFirstChild[1315], XMLData.GetNextChild[1316] to set the current child).

Errors
1500 The XMLData object is no longer valid.
1503 No iterator is initialized for this XMLData object.
1505 Invalid XMLData kind was specified.
1506 Invalid address for the return parameter was specified.
1507 Element cannot have Children
1512 Cyclic insertion - new data element is already part of document
1514 Invalid XMLData kind was specified for this position.
1900 Document must not be modified

25.3.2.29.19 InsertChildAfter

Method: InsertChildAfter (Node as XMLData, NewData as XMLData)

Description
Inserts a new XML node (supplied with the second parameter) after the specified node (first parameter).

Errors
1500 Invalid object.
1506 Invalid input xml
1507 No children allowed
1510 Invalid parameter.
1512 Child is already added
1514 Invalid kind at position

25.3.2.29.20 InsertChildBefore

Method: InsertChildBefore (Node as XMLData, NewData as XMLData)

Description

(C) 2015-2021 Altova GmbH

Inserts a new XML node (supplied with the second parameter) before the specified node (first parameter).

Errors

1500 Invalid object.
1506 Invalid input xml
1507 No children allowed
1510 Invalid parameter.
1512 Child is already added
1514 Invalid kind at position

25.3.2.29.21 IsSameNode

Declaration: IsSameNode (*pNodeToCompare* as XMLData[1309]) as Boolean

Description

Returns true if pNodeToCompare references the same node as the object itself.

Errors

1500 The XMLData object is no longer valid.
1506 Invalid address for the return parameter was specified.

25.3.2.29.22 Kind

Declaration: Kind as SPYXMLDataKind[1371]

Description
Kind of this XMLData object. This property is read-only.

Errors

1500 The XMLData object is no longer valid.
1510 Invalid address for the return parameter was specified.

25.3.2.29.23 MayHaveChildren

Declaration: MayHaveChildren as Boolean

Description
Indicates whether it is allowed to add children to this XMLData object.
This property is read-only.

Errors

1500 The XMLData object is no longer valid.
1510 Invalid address for the return parameter was specified.

25.3.2.29.24 Name

Declaration: Name as String

Description
Used to modify and to get the name of the XMLData object.

Errors
1500 The XMLData object is no longer valid.
1510 Invalid address for the return parameter was specified.

25.3.2.29.25 Parent

Declaration: Parent as XMLData 1309

Return value
Parent as XMLData object. Nothing (or NULL) if there is no parent element.

Description
Parent of this element. This property is read-only.

Errors
1500 The XMLData object is no longer valid.
1510 Invalid address for the return parameter was specified.

25.3.2.29.26 SetTextValueXMLEncoded

Method: SetTextValueXMLEncoded (*strVal* as String 1371)

Description
Sets the encoded text value of the XML.

Errors
1500 Invalid object.
1513 Modification not allowed.

25.3.2.29.27 TextValue

Declaration: TextValue as String

Description
Used to modify and to get the text value of this XMLData object.

Errors

(C) 2015-2021 Altova GmbH

1500 The XMLData object is no longer valid.
1510 Invalid address for the return parameter was specified.

25.3.3 Interfaces (obsolete)

Interfaces contained in this book are obsolete. It is recommended to migrate your applications to the new interfaces. See the different properties and methods in this book for migration hints.

25.3.3.1 AuthenticEvent (obsolete)

Superseded by AuthenticView 1185 and AuthenticRange 1157

The DocEditView object is renamed to OldAuthenticView.
DocEditSelection is renamed to AuthenticSelection.
DocEditEvent is renamed to AuthenticEvent.
DocEditDataTransfer is renamed to AuthenticDataTransfer.

Their usage - except for AuthenticDataTransfer - is no longer recommended. We will continue to support existing functionality for a yet undefined period of time but no new features will be added to these interface. All functionality available up to now in DocEditView 1335, DocEditSelection 1333, DocEditEvent 1321 and DocEditDataTransfer 1153 is now available via AuthenticView 1185, AuthenticRange 1157 and AuthenticDataTransfer 1152. Many new features have been added.

For examples on migrating from DocEdit to Authentic see the description of the different methods and properties of the different DocEdit objects.

Properties

altKey 1322
altLeft 1323
ctrlKey 1326
ctrlLeft 1327
shiftKey 1331
shiftLeft 1331

keyCode 1329
repeat 1330

button 1323

clientX 1325
clientY 1326

dataTransfer 1328

srcElement 1332
fromElement 1329

propertyName 1330

cancelBubble 1324
returnValue 1330

type 1333

Description
DocEditEvent interface.

25.3.3.1.1 altKey (obsolete)

Superseded by parameters to
AuthenticView.OnKeyboardEvent 1190 (On_AuthenticView_KeyPressed)
AuthenticView.OnMouseEvent 1191 (On_AuthenticView_MouseEvent)
AuthenticView.OnDragOver 1189 (On_AuthenticView_DragOver)

The event object that holds the information of the last event is now replaced by parameters to the different event handler functions to simplify data access. The event object will be supported for a not yet defined period of time for compatibility reasons. No improvements are planned. It is highly recommended to migrate to the new event handler functions.

```
// ----- XMLSpy scripting environment - javascript sample -----
// instead of:
// function On_DocEditKeyPressed ()
// {
//     if (Application.ActiveDocument.DocEditView.event.altKey ||
//         Application.ActiveDocument.DocEditView.event.altLeft)
//         MsgBox ("alt key is down");
// }
// use now:
function On_AuthenticView_KeyPressed (SPYKeyEvent i_eKeyEvent, long i_nKeyCode,
SPYVirtualKeyMask i_nVirtualKeyStatus)
{
    if (i_nVirtualKeyStatus & spyAltKeyMask)
        MsgBox ("alt key is down");
}
```

Declaration: altKey as Boolean

Description
True if the right ALT key is pressed.

(C) 2015-2021 Altova GmbH

25.3.3.1.2 altLeft (obsolete)

Superseded by parameters to
AuthenticView.OnKeyboardEvent[1190] (On_AuthenticView_KeyPressed)
AuthenticView.OnMouseEvent[1191] (On_AuthenticView_MouseEvent)
AuthenticView.OnDragOver[1189] (On_AuthenticView_DragOver)

The event object that holds the information of the last event is now replaced by parameters to the different event handler functions to simplify data access. The event object will be supported for a not yet defined period of time for compatibility reasons. No improvements are planned. It is highly recommended to migrate to the new event handler functions.

```
// ----- XMLSpy scripting environment - javascript sample -----
// instead of:
// function On_DocEditKeyDown ()
// {
//     if (Application.ActiveDocument.DocEditView.event.altKey ||
//         Application.ActiveDocument.DocEditView.event.altLeft)
//         MsgBox ("alt key is down");
// }
// use now:
function On_AuthenticView_KeyDown (SPYKeyEvent i_eKeyEvent, long i_nKeyCode, SPYVirtualKeyMask
i_nVirtualKeyStatus)
{
    if (i_nVirtualKeyStatus & spyAltKeyMask)
        MsgBox ("alt key is down");
}
```

Declaration: altLeft as Boolean

Description
True if the left ALT key is pressed.

25.3.3.1.3 button (obsolete)

Superseded by parameters to
AuthenticView.OnMouseEvent[1191] (On_AuthenticView_MouseEvent)
AuthenticView.OnDragOver[1189] (On_AuthenticView_DragOver)

The event object that holds the information of the last event is now replaced by parameters to the different event handler functions to simplify data access. The event object will be supported for a not yet defined period of time for compatibility reasons. No improvements are planned. It is highly recommended to migrate to the new event handler functions.

```
// ----- XMLSpy scripting environment - javascript sample -----
// instead of:
```

```
// function On_DocEditButtonDown ()
// {
//     if (Application.ActiveDocument.DocEditView.event.button == 1)
//         MsgBox ("left mouse button down detected");
// }
// use now:
function On_AuthenticView_MouseEvent (long i_nXPos, long i_nYPos, SPYMouseEvent i_eMouseEvent,
IAuthenticRange *i_ipRange)
{
    if (i_eMouseEvent & spyLeftButtonDownMask)
        MsgBox ("left mouse button down detected");
}
```

Declaration: button as long

Description
Specifies which mouse button is pressed:

0	No button is pressed.
1	Left button is pressed.
2	Right button is pressed.
3	Left and right buttons are both pressed.
4	Middle button is pressed.
5	Left and middle buttons both are pressed.
6	Right and middle buttons are both pressed.
7	All three buttons are pressed.

25.3.3.1.4 cancelBubble (obsolete)

Superseded by the boolean return value of following event handler functions
AuthenticView.OnKeyboardEvent 1190 (On_AuthenticView_KeyPressed)
AuthenticView.OnMouseEvent 1191 (On_AuthenticView_MouseEvent)
AuthenticView.OnDragOver 1189 (On_AuthenticView_DragOver)

The event object that holds the information of the last event is now replaced by parameters to the different event handler functions to simplify data access. The event object will be supported for a not yet defined period of time for compatibility reasons. No improvements are planned. It is highly recommended to migrate to the new event handler functions.

Returning *true* from an event handler function signals that the event has beend handled and normal event handling should be aborted.

```
// ----- XMLSpy scripting environment - javascript sample -----
// instead of:
// function On_DocEditKeyPressed ()
// {
//     if (Application.ActiveDocument.DocEditView.event.keyCode == 0x20)
```

```
//    {
//       // cancel key processing, swallow spaces :-)
//       Application.ActiveDocument.DocEditView.event.cancelBubble = true;
//    }
// }
// use now:
function On_AuthenticView_KeyPressed (SPYKeyEvent i_eKeyEvent, long i_nKeyCode,
SPYVirtualKeyMask i_nVirtualKeyStatus)
{
   if (i_nKeyCode == 0x20)
      return true;  // cancel key processing, swallow spaces :-)
}
```

Declaration: cancelBubble as Boolean

Description
Set cancelBubble to TRUE if the default event handler should not be called.

25.3.3.1.5 clientX (obsolete)

Superseded by parameters to
AuthenticView.OnMouseEvent 1191 (On_AuthenticView_MouseEvent)
AuthenticView.OnBeforeDrop 1187 (On_AuthenticView_BeforeDrop)
AuthenticView.OnDragOver 1189 (On_AuthenticView_DragOver)

The event object that holds the information of the last event is now replaced by parameters to the different event handler functions to simplify data access. The event object will be supported for a not yet defined period of time for compatibility reasons. No improvements are planned. It is highly recommended to migrate to the new event handler functions.

```
// ----- XMLSpy scripting environment - javascript sample -----
// instead of:
// function On_DocEditMouseMove ()
// {
//    MsgBox ("moving over " + Application.ActiveDocument.DocEditView.event.clientX +
//            "/" + Application.ActiveDocument.DocEditView.event.clientY);
// }
// use now:
function On_AuthenticView_MouseEvent (long i_nXPos, long i_nYPos, SPYMouseEvent i_eMouseEvent,
IAuthenticRange *i_ipRange)
{
   if (i_eMouseEvent & spyMouseMoveMask)
      MsgBox ("moving over " + i_nXPos + "/" + i_nYPos);
}
```

Declaration: clientX as long

Description
X value of the current mouse position in client coordinates.

25.3.3.1.6 clientY (obsolete)

Superseded by parameters to
AuthenticView.OnMouseEvent[1191] (On_AuthenticView_MouseEvent)
AuthenticView.OnBeforeDrop[1187] (On_AuthenticView_BeforeDrop)
AuthenticView.OnDragOver[1189] (On_AuthenticView_DragOver)

The event object that holds the information of the last event is now replaced by parameters to the different event handler functions to simplify data access. The event object will be supported for a not yet defined period of time for compatibility reasons. No improvements are planned. It is highly recommended to migrate to the new event handler functions.

```
// ----- XMLSpy scripting environment - javascript sample -----
// instead of:
// function On_DocEditMouseMove ()
// {
//     MsgBox ("moving over " + Application.ActiveDocument.DocEditView.event.clientX +
//             "/" + Application.ActiveDocument.DocEditView.event.clientY);
// }
// use now:
function On_AuthenticView_MouseEvent (long i_nXPos, long i_nYPos, SPYMouseEvent i_eMouseEvent,
IAuthenticRange *i_ipRange)
{
    if (i_eMouseEvent & spyMouseMoveMask)
        MsgBox ("moving over " + i_nXPos + "/" + i_nYPos);
}
```

Declaration: clientY as long

Description
Y value of the current mouse position in client coordinates.

25.3.3.1.7 ctrlKey (obsolete)

Superseded by parameters to
AuthenticView.OnKeyboardEvent[1190] (On_AuthenticView_KeyPressed)
AuthenticView.OnMouseEvent[1191] (On_AuthenticView_MouseEvent)
AuthenticView.OnDragOver[1189] (On_AuthenticView_DragOver)

(C) 2015-2021 Altova GmbH

The event object that holds the information of the last event is now replaced by parameters to the different event handler functions to simplify data access. The event object will be supported for a not yet defined period of time for compatibility reasons. No improvements are planned. It is highly recommended to migrate to the new event handler functions.

```
// ----- XMLSpy scripting environment - javascript sample -----
// instead of:
// function On_DocEditMouseMove ()
// {
//     if (Application.ActiveDocument.DocEditView.event.ctrlKey ||
//         Application.ActiveDocument.DocEditView.event.altLeft)
//         MsgBox ("control key is down");
// }
// use now:
function On_AuthenticView_MouseEvent (long i_nXPos, long i_nYPos, SPYMouseEvent i_eMouseEvent,
IAuthenticRange *i_ipRange)
{
    if (i_eMouseEvent & spyCtrlKeyMask)
        MsgBox ("control key is down");
}
```

Declaration: ctrlKey as Boolean

Description
True if the right CTRL key is pressed.

25.3.3.1.8 ctrlLeft (obsolete)

Superseded by parameters to
AuthenticView.OnKeyboardEvent[1190] (On_AuthenticView_KeyPressed)
AuthenticView.OnMouseEvent[1191] (On_AuthenticView_MouseEvent)
AuthenticView.OnDragOver[1189] (On_AuthenticView_DragOver)

The event object that holds the information of the last event is now replaced by parameters to the different event handler functions to simplify data access. The event object will be supported for a not yet defined period of time for compatibility reasons. No improvements are planned. It is highly recommended to migrate to the new event handler functions.

```
// ----- XMLSpy scripting environment - javascript sample -----
// instead of:
// function On_DocEditMouseMove ()
// {
//     if (Application.ActiveDocument.DocEditView.event.ctrlKey ||
//         Application.ActiveDocument.DocEditView.event.altLeft)
//         MsgBox ("control key is down");
// }
// use now:
```

```
function On_AuthenticView_MouseEvent (long i_nXPos, long i_nYPos, SPYMouseEvent i_eMouseEvent,
IAuthenticRange *i_ipRange)
{
    if (i_eMouseEvent & spyCtrlKeyMask)
        MsgBox ("control key is down");
}
```

Declaration: ctrlLeft as Boolean

Description
True if the left CTRL key is pressed.

25.3.3.1.9 dataTransfer (obsolete)

Superseded by parameters to
AuthenticView.OnBeforeDrop[1187] (On_AuthenticView_BeforeDrop)
AuthenticView.OnDragOver[1189] (On_AuthenticView_DragOver)

The event object that holds the information of the last event is now replaced by parameters to the different event handler functions to simplify data access. The event object will be supported for a not yet defined period of time for compatibility reasons. No improvements are planned. It is highly recommended to migrate to the new event handler functions.

```
// ----- XMLSpy scripting environment - javascript sample -----
// instead of:
// function On_DocEditDrop ()
// {
//     if (Application.ActiveDocument.DocEditView.event.dataTransfer != null)
//         if (! Application.ActiveDocument.DocEditView.event.dataTransfer.ownDrag)
//         {
//             // cancel key processing, don't drop foreign objects :-)
//             Application.ActiveDocument.DocEditView.event.cancelBubble = true;
//         }
// }
// use now:
function On_AuthenticView_BeforeDrop (long i_nXPos, long i_nYPos,
                                      IAuthenticRange *i_ipRange,
                                      IAuthenticDataTransfer *i_ipData)
{
    if (i_ipRange != null)
        if (! i_ipRange.ownDrag)
            return true;  // cancel key processing, don't drop foreign objects :-)

    return false;
}
```

Declaration: dataTransfer as Variant

(C) 2015-2021 Altova GmbH

Description
Property dataTransfer.

25.3.3.1.10 fromElement (obsolete)

Not supported

Declaration: fromElement as Variant (not supported)

Description
Currently no event sets this property.

25.3.3.1.11 keyCode (obsolete)

Superseded by a parameter to AuthenticView.OnKeyboardEvent 1190 (On_AuthenticView_KeyPressed)

The event object that holds the information of the last event is now replaced by parameters to the different event handler functions to simplify data access. The event object will be supported for a not yet defined period of time for compatibility reasons. No improvements are planned. It is highly recommended to migrate to the new event handler functions.

```
// ----- XMLSpy scripting environment - javascript sample -----
// instead of:
// function On_DocEditKeyPressed ()
// {
//     if (Application.ActiveDocument.DocEditView.event.keyCode == 0x20)
//     {
//         // cancel key processing, swallow spaces :-)
//         Application.ActiveDocument.DocEditView.event.cancelBubble = true;
//     }
// }
// use now:
function On_AuthenticView_KeyPressed (SPYKeyEvent i_eKeyEvent, long i_nKeyCode,
SPYVirtualKeyMask i_nVirtualKeyStatus)
{
    if (i_nKeyCode == 0x20)
        return true;  // cancel key processing, swallow spaces :-)
}
```

Declaration: keyCode as long

Description
Keycode of the currently pressed key. This property is read-write.

25.3.3.1.12 propertyName (obsolete)

Not supported

Declaration: propertyName as String (not supported)

Description
Currently no event sets this property.

25.3.3.1.13 repeat (obsolete)

Not supported

Declaration: repeat as Boolean (not supported)

Description
True if the onkeydown event is repeated.

25.3.3.1.14 returnValue (obsolete)

No longer supported

Declaration: returnValue as Variant

Description
Use returnValue to set a return value for your event handler.

(C) 2015-2021 Altova GmbH

25.3.3.1.15 shiftKey (obsolete)

Superseded by parameters to
AuthenticView.OnKeyboardEvent[1190] (On_AuthenticView_KeyPressed)
AuthenticView.OnMouseEvent[1191] (On_AuthenticView_MouseEvent)
AuthenticView.OnDragOver[1189] (On_AuthenticView_DragOver)

The event object that holds the information of the last event is now replaced by parameters to the different event handler functions to simplify data access. The event object will be supported for a not yet defined period of time for compatibility reasons. No improvements are planned. It is highly recommended to migrate to the new event handler functions.

```
// ----- XMLSpy scripting environment - javascript sample -----
// instead of:
// function On_DocEditDragOver ()
// {
//     if (Application.ActiveDocument.DocEditView.event.shiftKey ||
//         Application.ActiveDocument.DocEditView.event.shiftLeft)
//         MsgBox ("shift key is down");
// }
// use now:
function On_AuthenticView_DragOver (long i_nXPos, long i_nYPos,
                                    SPYMouseEvent i_eMouseEvent,
                                    IAuthenticRange *i_ipRange,
                                    IAuthenticDataTransfer *i_ipData)
{
   if (i_eMouseEvent & spyShiftKeyMask)
      MsgBox ("shift key is down");
}
```

Declaration: shiftKey as Boolean

Description
True if the right SHIFT key is pressed.

25.3.3.1.16 shiftLeft (obsolete)

Superseded by parameters to
AuthenticView.OnKeyboardEvent[1190] (On_AuthenticView_KeyPressed)
AuthenticView.OnMouseEvent[1191] (On_AuthenticView_MouseEvent)
AuthenticView.OnDragOver[1189] (On_AuthenticView_DragOver)

The event object that holds the information of the last event is now replaced by parameters to the different event handler functions to simplify data access. The event object will be supported for a not yet defined period of time for compatibility reasons. No improvements are planned. It is highly recommended to migrate to the new event handler functions.

```
// ----- XMLSpy scripting environment - javascript sample -----
// instead of:
// function On_DocEditDragOver ()
// {
//     if (Application.ActiveDocument.DocEditView.event.shiftKey ||
//         Application.ActiveDocument.DocEditView.event.shiftLeft)
//         MsgBox ("shift key is down");
// }
// use now:
function On_AuthenticView_DragOver (long i_nXPos, long i_nYPos,
                                    SPYMouseEvent i_eMouseEvent,
                                    IAuthenticRange *i_ipRange,
                                    IAuthenticDataTransfer *i_ipData)
{
   if (i_eMouseEvent & spyShiftKeyMask)
      MsgBox ("shift key is down");
}
```

Declaration: shiftLeft as Boolean

Description
True if the left SHIFT key is pressed.

25.3.3.1.17 srcElement (obsolete)

Superseded by parameters to
AuthenticView.OnMouseEvent 1191 (On_AuthenticView_MouseEvent)
AuthenticView.OnBeforeDrop 1187 (On_AuthenticView_BeforeDrop)
AuthenticView.OnDragOver 1189 (On_AuthenticView_DragOver)

The event object that holds the information of the last event is now replaced by parameters to the different event handler functions to simplify data access. The event object will be supported for a not yet defined period of time for compatibility reasons. No improvements are planned. It is highly recommended to migrate to the new event handler functions.

With the new event handler function, a range object selecting this element is provided instead of the XMLData element currently below the mouse cursor.

```
// ----- XMLSpy scripting environment - javascript sample -----
// instead of:
// function On_DocEditMouseMove ()
// {
//     var objEvent = Application.ActiveDocument.DocEditView.event;
//     if (objEvent.srcElement != null)
//         MsgBox ("moving over " + objEvent.srcElement.Parent.Name);
// }
```


(C) 2015-2021 Altova GmbH

```
// use now:
function On_AuthenticView_MouseEvent (long i_nXPos, long i_nYPos, SPYMouseEvent i_eMouseEvent,
IAuthenticRange *i_ipRange)
{
    if ((i_eMouseEvent & spyMouseMoveMask) &&
        (i_ipRange != null))
        MsgBox ("moving over " + i_ipRange.FirstXMLData.Parent.Name);
}
```

Declaration: srcElement as Variant

Description
Element which fires the current event. This is usually an XMLData[1309] object.

25.3.3.1.18 type (obsolete)

Not supported

Declaration: type as String (not supported)

Description
Currently no event sets this property.

25.3.3.2 AuthenticSelection (obsolete)

Superseded by AuthenticRange[1157]

The DocEditView object is renamed to OldAuthenticView.
DocEditSelection is renamed to AuthenticSelection.
DocEditEvent is renamed to AuthenticEvent.
DocEditDataTransfer is renamed to AuthenticDataTransfer.

Their usage - except for AuthenticDataTransfer - is no longer recommended. We will continue to support existing functionality for a yet undefined period of time but no new features will be added to these interface. All functionality available up to now in DocEditView[1335], DocEditSelection[1333], DocEditEvent[1321] and DocEditDataTransfer[1152] is now available via AuthenticView[1185], AuthenticRange[1157] and AuthenticDataTransfer[1152]. Many new features have been added.

For examples on migrating from DocEdit to Authentic see the description of the different methods and properties of the different DocEdit objects.

Properties
Start 1335
StartTextPosition 1335
End 1334
EndTextPosition 1334

25.3.3.2.1 End (obsolete)

Superseded by AuthenticRange.LastXMLData 1176

```
// ----- javascript sample -----
// instead of:
// var objXMLData = Application.ActiveDocument.DocEditView.CurrentSelection.End;
// use now:
var objXMLData = Application.ActiveDocument.AuthenticView.Selection.LastXMLData;
```

Declaration: End as XMLData 1309

Description
XML element where the current selection ends.

25.3.3.2.2 EndTextPosition (obsolete)

Superseded by AuthenticRange.LastXMLDataOffset 1177

```
// ----- javascript sample -----
// instead of:
// var nOffset = Application.ActiveDocument.DocEditView.CurrentSelection.EndTextPosition;
// use now:
var nOffset = Application.ActiveDocument.AuthenticView.Selection.LastXMLDataOffset;
```

Declaration: EndTextPosition as long

Description
Position in DocEditSelection.End.TextValue 1334 where the selection ends.

(C) 2015-2021 Altova GmbH

25.3.3.2.3 Start (obsolete)

Superseded by AuthenticRange.FirstXMLData 1164

```
// ----- javascript sample -----
// instead of:
// var objXMLData = Application.ActiveDocument.DocEditView.CurrentSelection.Start;
// use now:
var objXMLData = Application.ActiveDocument.AuthenticView.Selection.FirstXMLData;
```

Declaration: Start as XMLData 1309

Description
XML element where the current selection starts.

25.3.3.2.4 StartTextPosition (obsolete)

Superseded by AuthenticRange.FirstXMLDataOffset 1165

```
// ----- javascript sample -----
// instead of:
// var nOffset = Application.ActiveDocument.DocEditView.CurrentSelection.StartTextPosition;
// use now:
var nOffset = Application.ActiveDocument.AuthenticView.Selection.FirstXMLDataOffset;
```

Declaration: StartTextPosition as long

Description
Position in DocEditSelection.Start.TextValue 1335 where the selection starts.

25.3.3.3 OldAuthentictView (obsolete)

Superseded by AuthenticView 1185 and **AuthenticRange** 1157

The DocEditView object is renamed to OldAuthenticView.
DocEditSelection is renamed to AuthenticSelection.
DocEditEvent is renamed to AuthenticEvent.
DocEditDataTransfer is renamed to AuthenticDataTransfer.

Their usage - except for AuthenticDataTransfer - is no longer recommended. We will continue to support existing functionality for a yet undefined period of time but no new features will be added to these interfaces. All functionality available up to now in DocEditView 1335, DocEditSelection 1333, DocEditEvent 1321 and

DocEditDataTransfer 1153 is now available via AuthenticView 1185, AuthenticRange 1157 and AuthenticDataTransfer 1152. Many new features have been added.

For examples on migrating from DocEdit to Authentic see the description of the different methods and properties of the different DocEdit objects.

Methods

LoadXML 1352
SaveXML 1356

EditClear 1338
EditCopy 1339
EditCut 1339
EditPaste 1340
EditRedo 1340
EditSelectAll 1340
EditUndo 1341

RowAppend 1353
RowDelete 1353
RowDuplicate 1354
RowInsert 1354
RowMoveDown 1355
RowMoveUp 1355

ApplyTextState 1337
IsTextStateApplied 1351
IsTextStateEnabled 1351

MarkUpView 1352

SelectionSet 1357
SelectionMoveTabOrder 1356

GetNextVisible 1344
GetPreviousVisible 1344

GetAllowedElements 1342

Properties

CurrentSelection 1338

event 1341

XMLRoot 1358

IsEditClearEnabled 1345
IsEditCopyEnabled 1345
IsEditCutEnabled 1346
IsEditPasteEnabled 1346
IsEditRedoEnabled 1347
IsEditUndoEnabled 1347

(C) 2015-2021 Altova GmbH

IsRowAppendEnabled 1348
IsRowDeleteEnabled 1348
IsRowDuplicateEnabled 1349
IsRowInsertEnabled 1349
IsRowMoveDownEnabled 1350
IsRowMoveUpEnabled 1350

Description
Interface for Authentic View.

25.3.3.3.1 ApplyTextState (obsolete)

Superseded by AuthenticRange.PerformAction 1180

Use spyAuthenticApply for the eAction parameter. The PerformAction method allows to apply text state attributes to any range of the document, not only the current UI selection.

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.DocEditView.ApplyTextState ("bold");
// use now:
if (! Application.ActiveDocument.AuthenticView.Selection.PerformAction (spyAuthenticApply, "bold"))
    MsgBox ("Error: can't set current selection to bold");
```

Declaration: ApplyTextState (*elementName* as String)

Description
Applies or removes the text state defined by the parameter elementName. Common examples for the parameter elementName would be strong and italic.

In an XML document there are segments of data, which may contain sub-elements. For example consider the following HTML:

```
<b>fragment</b>
```

The HTML tag <b> will cause the word fragment to be bold. However, this only happens because the HTML parser knows that the tag <b> is bold. With XML there is much more flexibility. It is possible to define any XML tag to do anything you desire. The point is that it is possible to apply a Text state using XML. But the Text state that is applied must be part of the schema. For example in the OrgChart.xml, OrgChart.sps, OrgChart.xsd example the tag is the same as bold. And to apply bold the method ApplyTextState() is called. But like the row and edit operations it is necessary to test if it is possible to apply the text state.

See also IsTextStateEnabled 1351 and IsTextStateApplied 1351.

25.3.3.3.2 CurrentSelection (obsolete)

Superseded by AuthenticView.Selection 1201

The returned AuthenticRange 1157 object supports navigation via XMLData elements as well as navigation by document elements (e.g. characters, words, tags) or text cursor positions.

```
// ----- javascript sample -----
// instead of:
// var objDocEditSel = Application.ActiveDocument.DocEditView.CurrentSelection;
// use now:
    var objRange = Application.ActiveDocument.AuthenticView.Selection;
```

Declaration: CurrentSelection as DocEditSelection 1333

Description
The property provides access to the current selection in the Authentic View.

25.3.3.3.3 EditClear (obsolete)

Superseded by AuthenticRange.Delete 1161

The Delete method of AuthenticRange allows to delete any range of the document, not only the current UI selection.

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.DocEditView.EditClear();
// use now:
if (! Application.ActiveDocument.AuthenticView.Selection.Delete())
    MsgBox ("Error: can't delete current selection");
```

Declaration: EditClear

Description
Deletes the current selection.

(C) 2015-2021 Altova GmbH

25.3.3.3.4 EditCopy (obsolete)

Superseded by AuthenticRange.Copy 1161

The Copy method of AuthenticRange allows to delete any range of the document, not only the current UI selection.

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.DocEditView.EditCopy();
// use now:
if (! Application.ActiveDocument.AuthenticView.Selection.Copy())
    MsgBox ("Error: can't copy current selection");
```

Declaration: EditCopy

Description
Copies the current selection to the clipboard.

25.3.3.3.5 EditCut (obsolete)

Superseded by AuthenticRange.Cut 1161

The Cut method of AuthenticRange allows to delete any range of the document, not only the current UI selection.

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.DocEditView.EditCut();
// use now:
if (! Application.ActiveDocument.AuthenticView.Selection.Cut())
    MsgBox ("Error: can't cut out current selection");
```

Declaration: EditCut

Description
Cuts the current selection from the document and copies it to the clipboard.

25.3.3.3.6 EditPaste (obsolete)

Superseded by AuthenticRange.Paste 1179

The Paste method of AuthenticRange allows to delete any range of the document, not only the current UI selection.

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.DocEditView.EditPaste();
// use now:
if (! Application.ActiveDocument.AuthenticView.Selection.Paste())
    MsgBox ("Error: can't paste to current selection");
```

Declaration: EditPaste

Description
Pastes the content from the clipboard into the document.

25.3.3.3.7 EditRedo (obsolete)

Superseded by AuthenticView.Redo 1201

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.DocEditView.EditRedo();
// use now:
if (! Application.ActiveDocument.AuthenticView.Redo())
    MsgBox ("Error: no redo step available");
```

Declaration: EditRedo

Description
Redo the last undo step.

25.3.3.3.8 EditSelectAll (obsolete)

Superseded by AuthenticView.WholeDocument 1203 and **AuthenticRange.Select** 1181

(C) 2015-2021 Altova GmbH

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.DocEditView.EditSelectAll();
// use now:
Application.ActiveDocument.AuthenticView.WholeDocument.Select();
```

Declaration: EditSelectAll

Description
The method selects the complete document.

25.3.3.3.9 EditUndo (obsolete)

Superseded by AuthenticView.Undo 1202

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.DocEditView.EditUndo();
// use now:
if (! Application.ActiveDocument.AuthenticView.Undo())
    MsgBox ("Error: no undo step available");
```

Declaration: EditUndo

Description
Undo the last action.

25.3.3.3.10 event (obsolete)

Superseded by parameters to AuthenticView events 1185**.**

Declaration: event as DocEditEvent 1321

Description
The event property holds a DocEditEvent object which contains information about the current event.

25.3.3.3.11 GetAllowedElements (obsolete)

Superseded by AuthenticRange.CanPerformActionWith 1160

AuthenticRange now supports all functionality of the 'elements' entry helper. Besides querying the elements that can be inserted, appended, etc., you can invoke the action as well. See AuthenticRange.PerformAction 1180 for more information.

```
// ----- javascript sample -----
// instead of:
// var arrElements = New Array();
// var objDocEditView = Application.ActiveDocument.DocEditView;
// var objStartElement = objDocEditView.CurrentSelection.Start;
// var objEndElement = objDocEditView.CurrentSelection.End;
// objDocEditView.GetAllowedElements(k_ActionInsertBefore, objStartElement, objEndElement,
arrElements);
// use now:
var arrElements = New Array();
Application.ActiveDocument.AuthenticView.Selection.CanPerformActionWith (spyAuthenticInsertBefore,
arrElements);
```

Declaration: GetAllowedElements (*nAction* as SpyAuthenticElementActions 1359, *pStartElement* as XMLData 1309, *pEndElement* as XMLData 1309, *pElements* as Variant)

Description
GetAllowedElements() returns the allowed elements for the various actions specified by nAction.

JavaScript example:

```
Function GetAllowed()
{
        var objView = Application.ActiveDocument.DocEditView;

        var arrElements = New Array(1);

        var objStart = objView.CurrentSelection.Start;
        var objEnd = objView.CurrentSelection.End;

        var strText;
        strText = "valid elements at current selection:\n\n";

        For(var i = 1;i <= 4;i++) {
                objPlugIn.GetAllowedElements(i,objStart,objEnd,arrElements);
                strText = strText + ListArray(arrElements) + "------------------\n";
        }

        Return strText;
}

Function ListArray(arrIn)
{
        var strText = "";
```

```
        If(TypeOf(arrIn) == "object")      {
                For(var i = 0;i <= (arrIn.length - 1);i++)
                        strText = strText + arrIn[i] + "\n";
        }

        Return strText;
}
```

VBScript example:

```
Sub DisplayAllowed
        Dim objView
        Set objView = Application.ActiveDocument.DocEditView

        Dim arrElements()

        Dim objStart
        Dim objEnd
        Set objStart = objView.CurrentSelection.Start
        Set objEnd = objView.CurrentSelection.End

        Dim strText
        strText = "valid elements at current selection:" & chr(13) & chr(13)

        Dim i

        For i = 1 To 4
                objView.GetAllowedElements i,objStart,objEnd,arrElements
                strText = strText & ListArray(arrElements) & "---------------" & chr(13)
        Next

        msgbox strText
End Sub

Function ListArray(arrIn)
        Dim strText

        If IsArray(arrIn) Then
                Dim i

                For i = 0 To UBound(arrIn)
                        strText = strText & arrIn(i) & chr(13)
                Next
        End If

        ListArray = strText
End Function
```

25.3.3.3.12 GetNextVisible (obsolete)

Superseded by AuthenticRange.SelectNext 1181

AuthenticRange now supports a wide range of element navigation methods based on document elements like characters, words, tags and many more. Selecting the text passage that represents the content of the next XML element is just one of them.

```
// ----- javascript sample -----
// instead of:
// var objCurrXMLData = ...
// var objXMLData = Application.ActiveDocument.DocEditView.GetNextVisible(objCurrXMLData);
// Application.ActiveDocument.DocEditView.SelectionSet (objXMLData, 0, objXMLData, -1);
// use now:
var objRange = ...
try
   { objRange.SelectNext (spyAuthenticTag).Select(); }
catch (err)
{
   if ((err.number & 0xffff) == 2003)
      MsgBox ("end of document reached");
   else
      throw (err);
}
```

Declaration: GetNextVisible (*pElement* as XMLData 1309) as XMLData 1309

Description
The method gets the next visible XML element in the document.

25.3.3.3.13 GetPreviousVisible (obsolete)

Superseded by AuthenticRange.SelectPrevious 1182

AuthenticRange now supports a wide range of element navigation methods based on document elements like characters, words, tags and many more. Selecting the text passage that represents the content of the previous XML element is just one of them.

```
// ----- javascript sample -----
// instead of:
// var objCurrXMLData = ...
// var objXMLData = Application.ActiveDocument.DocEditView.GetPreviousVisible(objCurrXMLData);
// Application.ActiveDocument.DocEditView.SelectionSet (objXMLData, 0, objXMLData, -1);
// use now:
var objRange = ...
try
```


(C) 2015-2021 Altova GmbH

```
    { objRange.SelectPrevious (spyAuthenticTag).Select(); }
catch (err)
{
    if ((err.number & 0xffff) == 2004)
        MsgBox ("begin of document reached");
    else
        throw (err);
}
```

Declaration: GetPreviousVisible (*pElement* as XMLData[1309]) as XMLData[1309]

Description
The method gets the previous visible XML element in the document.

25.3.3.3.14 IsEditClearEnabled (obsolete)

Superseded by AuthenticRange.IsDeleteEnabled[1173]

The IsDeleteEnabled property is now supported for any range of the document, not only the current UI selection.

```
// ----- javascript sample -----
// instead of:
// if (Application.ActiveDocument.DocEditView.IsEditClearEnabled)
//     Application.ActiveDocument.DocEditView.EditClear();
// use now:
var objCurrSelection = Application.ActiveDocument.AuthenticView.Selection;
if (objCurrSelection.IsDeleteEnabled)
    objCurrSelection.Delete();
```

Declaration: IsEditClearEnabled as Boolean

Description
True if EditClear[1338] is possible. See also Editing operations[1129].

25.3.3.3.15 IsEditCopyEnabled (obsolete)

Superseded by AuthenticRange.IsCopyEnabled[1173]

The IsCopyEnabled property is now supported for any range of the document, not only the current UI selection.

```
// ----- javascript sample -----
// instead of:
// if (Application.ActiveDocument.DocEditView.IsEditCopyEnabled)
//    Application.ActiveDocument.DocEditView.EditCopy();
// use now:
var objCurrSelection = Application.ActiveDocument.AuthenticView.Selection;
if (objCurrSelection.IsCopyEnabled)
   objCurrSelection.Copy();
```

Declaration: IsEditCopyEnabled as Boolean

Description
True if copy to clipboard is possible. See also EditCopy[1339] and Editing operations[1129].

25.3.3.3.16 IsEditCutEnabled (obsolete)

Superseded by AuthenticRange.IsCutEnabled[1173]

The IsCutEnabled property is now supported for any range of the document, not only the current UI selection.

```
// ----- javascript sample -----
// instead of:
// if (Application.ActiveDocument.DocEditView.IsEditCutEnabled)
//    Application.ActiveDocument.DocEditView.EditCut();
// use now:
var objCurrSelection = Application.ActiveDocument.AuthenticView.Selection;
if (objCurrSelection.IsCutEnabled)
   objCurrSelection.Cut();
```

Declaration: IsEditCutEnabled as Boolean

Description
True if EditCut[1339] is currently possible. See also Editing operations[1129].

25.3.3.3.17 IsEditPasteEnabled (obsolete)

Superseded by AuthenticRange.IsPasteEnabled[1175]

(C) 2015-2021 Altova GmbH

The IsPasteEnabled property is now supported for any range of the document, not only the current UI selection.

```
// ----- javascript sample -----
// instead of:
// if (Application.ActiveDocument.DocEditView.IsEditPasteEnabled)
//    Application.ActiveDocument.DocEditView.EditPaste();
// use now:
var objCurrSelection = Application.ActiveDocument.AuthenticView.Selection;
if (objCurrSelection.IsPasteEnabled)
   objCurrSelection.Paste();
```

Declaration: IsEditPasteEnabled as Boolean

Description
True if EditPaste 1340 is possible. See also Editing operations 1129.

25.3.3.3.18 IsEditRedoEnabled (obsolete)

Superseded by AuthenticView.IsRedoEnabled 1199

```
// ----- javascript sample -----
// instead of:
// if (Application.ActiveDocument.DocEditView.IsEditRedoEnabled)
//    Application.ActiveDocument.DocEditView.EditRedo();
// use now:
if (Application.ActiveDocument.AuthenticView.IsRedoEnabled)
   Application.ActiveDocument.AuthenticView.Redo();
```

Declaration: IsEditRedoEnabled as Boolean

Description
True if EditRedo 1340 is currently possible. See also Editing operations 1129.

25.3.3.3.19 IsEditUndoEnabled (obsolete)

Superseded by AuthenticView.IsUndoEnabled 1199

```
// ----- javascript sample -----
// instead of:
// if (Application.ActiveDocument.DocEditView.IsEditUndoEnabled)
```

```
//    Application.ActiveDocument.DocEditView.EditUndo();
// use now:
if (Application.ActiveDocument.AuthenticView.IsUndoEnabled)
    Application.ActiveDocument.AuthenticView.Undo();
```

Declaration: IsEditUndoEnabled as Boolean

Description
True if EditUndo[1341] is possible. See also Editing operations[1129].

25.3.3.3.20 IsRowAppendEnabled (obsolete)

Superseded by AuthenticRange.IsInDynamicTable[1174]

The operations 'insert', 'append', 'delete' and 'duplicate' row are available whenever the selection is inside a dynamic table.

```
// ----- javascript sample -----
// instead of:
// if (Application.ActiveDocument.DocEditView.IsRowAppendEnabled)
//    Application.ActiveDocument.DocEditView.RowAppend();
// use now:
if (Application.ActiveDocument.AuthenticView.Selection.IsInDynamicTable())
    Application.ActiveDocument.AuthenticView.Selection.AppendRow();
```

Declaration: IsRowAppendEnabled as Boolean

Description
True if RowAppend[1353] is possible. See also Row operations[1128].

25.3.3.3.21 IsRowDeleteEnabled (obsolete)

Superseded by AuthenticRange.IsInDynamicTable[1174]

The operations 'insert', 'append', 'delete' and 'duplicate' row are available whenever the selection is inside a dynamic table.

```
// ----- javascript sample -----
// instead of:
```


(C) 2015-2021 Altova GmbH

```
// if (Application.ActiveDocument.DocEditView.IsRowDeleteEnabled)
//    Application.ActiveDocument.DocEditView.Rowdelete();
// use now:
if (Application.ActiveDocument.AuthenticView.Selection.IsInDynamicTable())
   Application.ActiveDocument.AuthenticView.Selection.DeleteRow();
```

Declaration: IsRowDeleteEnabled as Boolean

Description
True if RowDelete[1353] is possible. See also Row operations[1128].

25.3.3.3.22 IsRowDuplicateEnabled (obsolete)

Superseded by AuthenticRange.IsInDynamicTable[1174]

The operations 'insert', 'append', 'delete' and 'duplicate' row are available whenever the selection is inside a dynamic table.

```
// ----- javascript sample -----
// instead of:
// if (Application.ActiveDocument.DocEditView.IsRowDuplicateEnabled)
//    Application.ActiveDocument.DocEditView.RowDuplicate();
// use now:
if (Application.ActiveDocument.AuthenticView.Selection.IsInDynamicTable())
   Application.ActiveDocument.AuthenticView.Selection.DuplicateRow();
```

Declaration: IsRowDuplicateEnabled as Boolean

Description
True if RowDuplicate[1354] is currently possible. See also Row operations[1128].

25.3.3.3.23 IsRowInsertEnabled (obsolete)

Superseded by AuthenticRange.IsInDynamicTable[1174]

The operations 'insert', 'append', 'delete' and 'duplicate' row are available whenever the selection is inside a dynamic table.

```
// ----- javascript sample -----
// instead of:
// if (Application.ActiveDocument.DocEditView.IsRowInsertEnabled)
```

```
//    Application.ActiveDocument.DocEditView.RowInsert();
// use now:
if (Application.ActiveDocument.AuthenticView.Selection.IsInDynamicTable())
    Application.ActiveDocument.AuthenticView.Selection.InsertRow();
```

Declaration: IsRowInsertEnabled as Boolean

Description
True if RowInsert[1354] is possible. See also Row operations[1128].

25.3.3.3.24 IsRowMoveDownEnabled (obsolete)

Superseded by AuthenticRange.IsLastRow[1174]

```
// ----- javascript sample -----
// instead of:
// if (Application.ActiveDocument.OldAuthenticView.IsRowMoveDownEnabled)
//    Application.ActiveDocument.DocEditView.RowMoveDown();
// use now:
if (!Application.ActiveDocument.AuthenticView.Selection.IsLastRow)
    Application.ActiveDocument.AuthenticView.Selection.MoveRowDown();
```

Declaration: IsRowMoveDownEnabled as Boolean

Description
True if RowMoveDown[1355] is currently possible. See also Row operations[1128].

25.3.3.3.25 IsRowMoveUpEnabled (obsolete)

Superseded by AuthenticRange.IsFirstRow[1174]

```
// ----- javascript sample -----
// instead of:
// if (Application.ActiveDocument.DocEditView.IsRowMoveUpEnabled)
//    Application.ActiveDocument.DocEditView.RowMoveUp();
// use now:
if (!Application.ActiveDocument.AuthenticView.Selection.IsFirstRow)
    Application.ActiveDocument.AuthenticView.Selection.MoveRowUp();
```


(C) 2015-2021 Altova GmbH

Declaration: IsRowMoveUpEnabled as Boolean

Description
True if RowMoveUp 1355 is possible. See also Row operations 1128.

25.3.3.3.26 IsTextStateApplied (obsolete)

Superseded by AuthenticRange.IsTextStateApplied 1175

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.DocEditView.IsTextStateApplied ("bold");
// use now:
if (Application.ActiveDocument.AuthenticView.Selection.IsTextStateApplied ("bold"))
    MsgBox ("bold on");
else
    MsgBox ("bold off");
```

Declaration: IsTextStateApplied (*elementName* as String) as Boolean

Description
Checks to see if the it the text state has already been applied. Common examples for the parameter elementName would be strong and italic.

25.3.3.3.27 IsTextStateEnabled (obsolete)

Superseded by AuthenticRange.CanPerformAction 1159

Use spyAuthenticApply for the eAction parameter. The CanPerformAction method allows to operate on any range of the document, not only the current UI selection.

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.DocEditView.IsTextStateEnabled ("bold");
// use now:
if (Application.ActiveDocument.AuthenticView.Selection.CanPerformAction (spyAuthenticApply, "bold"))
    ... // e.g. enable 'bold' button
```

Declaration: IsTextStateEnabled (*i_strElementName* as String) as Boolean

Description

Checks to see if it is possible to apply a text state. Common examples for the parameter elementName would be strong and italic.

25.3.3.3.28 LoadXML (obsolete)

Superseded by AuthenticView.AsXMLString 1195

AuthenticView now supports the property AsXMLString that can be used to directly access and replace the document content as an XMLString.

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.DocEditView.LoadXML (strDocAsXMLString);
// use now:
try
   { Application.ActiveDocument.AuthenticView.AsXMLString = strDocAsXMLString; }
catch (err)
   { MsgBox ("Error: invalid XML string"); }
```

Declaration: LoadXML (*xmlString* as String)

Description
Loads the current XML document with the XML string applied. The new content is displayed immediately. The *xmlString* parameter must begin with the XML declaration, e.g.,
objPlugIn.LoadXML("<?xml version='1.0' encoding='UTF-8'?><root></root>");

25.3.3.3.29 MarkUpView (obsolete)

Superseded by AuthenticView.MarkupVisibility 1200

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.DocEditView.MarkuUpView = 2;
// use now:
Application.ActiveDocument.AuthenticView.MarkupVisibility = spyAuthenticMarkupLarge;
```

Declaration: MarkUpView (*kind* as long)

Description
By default the document displayed is using HTML techniques. But sometimes it is desirable to show the editing tags. Using this method it is possible to display three different types of markup tags:

 (C) 2015-2021 Altova GmbH

0 hide the markup tags
2 show the large markup tags
3 show the mixed markup tags.

25.3.3.3.30 RowAppend (obsolete)

Superseded by AuthenticRange.AppendRow 1158

The table operations of AuthenticRange now allow to manipulate any table in the current document independent of the current UI selection.

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.DocEditView.RowAppend();
// use now:
if (! Application.ActiveDocument.AuthenticView.Selection.AppendRow())
    MsgBox ("Error: can't append row");
```

Declaration: RowAppend

Description
Appends a row at the current position.

See also Row operations 1128.

25.3.3.3.31 RowDelete (obsolete)

Superseded by AuthenticRange.DeleteRow 1162

The table operations of AuthenticRange now allow to manipulate any table in the current document independent of the current UI selection.

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.DocEditView.RowDelete();
// use now:
if (! Application.ActiveDocument.AuthenticView.Selection.DeleteRow())
    MsgBox ("Error: can't delete row");
```

Declaration: RowDelete

Description
Deletes the currently selected row(s).

See also Row operations 1128.

25.3.3.3.32 RowDuplicate (obsolete)

Superseded by AuthenticRange.DuplicateRow 1162

The table operations of AuthenticRange now allow to manipulate any table in the current document independent of the current UI selection.

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.DocEditView.RowDuplicate();
// use now:
if (! Application.ActiveDocument.AuthenticView.Selection.DuplicateRow())
    MsgBox ("Error: can't duplicate row");
```

Declaration: RowDuplicate

Description
The method duplicates the currently selected rows.

See also Row operations 1128.

25.3.3.3.33 RowInsert (obsolete)

Superseded by AuthenticRange.InsertRow 1172

The table operations of AuthenticRange now allow to manipulate any table in the current document independent of the current UI selection.

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.DocEditView.RowInsert();
// use now:
if (! Application.ActiveDocument.AuthenticView.Selection.InsertRow())
    MsgBox ("Error: can't insert row");
```

Declaration: RowInsert

(C) 2015-2021 Altova GmbH

Description
Inserts a new row immediately above the current selection.

See also Row operations 1128.

25.3.3.3.34 RowMoveDown (obsolete)

Superseded by AuthenticRange.MoveRowDown 1179

The table operations of AuthenticRange now allow to manipulate any table in the current document independent of the current UI selection.

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.DocEditView.RowMoveDown();
// use now:
if (! Application.ActiveDocument.AuthenticView.Selection.MoveRowDown())
    MsgBox ("Error: can't move row down");
```

Declaration: RowMoveDown

Description
Moves the current row one position down.

See also Row operations 1128.

25.3.3.3.35 RowMoveUp (obsolete)

Superseded by AuthenticRange.MoveRowUp 1179

The table operations of AuthenticRange now allow to manipulate any table in the current document independent of the current UI selection.

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.DocEditView.RowAppend();
// use now:
if (! Application.ActiveDocument.AuthenticView.Selection.MoveRowUp())
    MsgBox ("Error: can't move row up");
```

Declaration: RowMoveUp

Description
Moves the current row one position up.

See also Row operations 1128.

25.3.3.3.36 SaveXML (obsolete)

Superseded by AuthenticView.AsXMLString 1195

AuthenticView now supports the property XMLString that can be used to directly access and replace the document content as an XMLString.

```
// ----- javascript sample -----
// instead of:
// var strDocAsXMLString = Application.ActiveDocument.DocEditView.SaveXML();
// use now:
try
{
    var strDocAsXMLString = Application.ActiveDocument.AuthenticView.AsXMLString;
    ... // do something here
}
catch (err)
    { MsgBox ("Error: invalid XML string"); }
```

Declaration: SaveXML as String

Return Value
XML structure as string

Description
Saves the current XML data to a string that is returned to the caller.

25.3.3.3.37 SelectionMoveTabOrder (obsolete)

Superseded by AuthenticRange.SelectNext 1181

AuthenticRange now supports a wide range of element navigation methods based on document elements like characters, words, tags and many more. Selecting the next paragraph is just one of them, and navigation is not necessarily bound to the current UI selection.

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.DocEditView.SelectionMoveTabOrder(true, true);
```


(C) 2015-2021 Altova GmbH

```
// use now:
Application.ActiveDocument.AuthenticView.Selection.SelectNext (spyAuthenticParagraph).Select();
// to append a row to a table use AuthenticRange.AppendRow
```

Declaration: SelectionMoveTabOrder (*bForward* as Boolean, *bTag* as Boolean)

Description
SelectionMoveTabOrder() moves the current selection forwards or backwards.

If bTag is false and the current selection is at the last cell of a table a new line will be added.

25.3.3.3.38 SelectionSet (obsolete)

Superseded by AuthenticRange.FirstXMLData 1164 and related properties

AuthenticRange supports navigation via XMLData elements as well as navigation by document elements (e.g. characters, words, tags) or text cursor positions.

```
// ----- javascript sample -----
// instead of:
// if (! Application.ActiveDocument.DocEditView.SelectionSet(varXMLData1, 0, varXMLData2, -1))
//    MsgBox ("Error: invalid data position");
// use now:
try
{
    var objSelection = Application.ActiveDocument.AuthenticView.Selection;
    objSelection.FirstXMLData = varXMLData1;
    objSelection.FirstXMLdataOffset = 0;
    objSelection.LastXMLData = varXMLData2;
    objSelection.LastXMLDataOffset = -1;
    objSelection.Select();
}
catch (err)
    { MsgBox ("Error: invalid data position"); }
// to select all text between varXMLData1 and varXMLdata2, inclusive
```

Declaration: SelectionSet (*pStartElement* as XMLData 1309, *nStartPos* as long, *pEndElement* as XMLData 1309, *nEndPos* as long) as Boolean

Description
Use SelectionSet() to set a new selection in the Authentic View. Its possible to set pEndElement to null (nothing) if the selection should be just over one (pStartElement) XML element.

25.3.3.3.39 XMLRoot (obsolete)

Superseded by AuthenticView.XMLDataRoot 1203

```
// ----- javascript sample -----
// instead of:
// var objXMLData = Application.ActiveDocument.DocEditView.XMLRoot;
// use now:
var objXMLData = Application.ActiveDocument.AuthenticView.XMLDataRoot;
```

Declaration: XMLRoot as XMLData 1309

Description
XMLRoot is the parent element of the currently displayed XML structure. Using the XMLData 1309 interface you have full access to the complete content of the file.

25.3.4 Enumerations

This is a list of all enumerations used by the XMLSpy API. If your scripting environment does not support enumerations use the number-values instead.

25.3.4.1 ENUMApplicationStatus

Description
Enumeration to specify the current Application status.

Possible values:

eApplicationRunning	= 0
eApplicationAfterLicenseCheck	= 1
eApplicationBeforeLicenseCheck	= 2
eApplicationConcurrentLicenseCheckFailed	= 3
eApplicationProcessingCommandLine	= 4

25.3.4.2 SPYAttributeTypeDefinition

Description
Attribute type definition that can be selected for generation of Sample XML.
This type is used with the method GenerateDTDOrSchema 1233 and GenerateDTDOrSchemaEx 1233.

Possible values:

spyMergedGlobal	= 0
spyDistinctGlobal	= 1

spyLocal = 2

25.3.4.3 SPYAuthenticActions

Description
Actions that can be performed on AuthenticRange[1157] objects.

Possible values:

spyAuthenticInsertAt = 0
spyAuthenticApply = 1
spyAuthenticClearSurr = 2
spyAuthenticAppend = 3
spyAuthenticInsertBefore = 4
spyAuthenticRemove = 5

25.3.4.4 SPYAuthenticDocumentPosition

Description
Relative and absolute positions used for navigating with AuthenticRange[1157] objects.

Possible values:

spyAuthenticDocumentBegin = 0
spyAuthenticDocumentEnd = 1
spyAuthenticRangeBegin = 2
spyAuthenticRangeEnd = 3

25.3.4.5 SPYAuthenticElementActions

Description
Actions that can be used with the obsolete object GetAllowedElements (superseded by AuthenticRange.CanPerformActionWith[1160]).

Possible values:

k_ActionInsertAt = 0
k_ActionApply = 1
k_ActionClearSurr = 2
k_ActionAppend = 3
k_ActionInsertBefore = 4
k_ActionRemove = 5

25.3.4.6 SPYAuthenticElementKind

Description
Enumeration of the different kinds of elements used for navigation and selection within the AuthenticRange[1157] and AuthenticView[1185] objects.

Possible values:

spyAuthenticChar	= 0
spyAuthenticWord	= 1
spyAuthenticLine	= 3
spyAuthenticParagraph	= 4
spyAuthenticTag	= 6
spyAuthenticDocument	= 8
spyAuthenticTable	= 9
spyAuthenticTableRow	= 10
spyAuthenticTableColumn	= 11

25.3.4.7 SPYAuthenticMarkupVisibility

Description
Enumeration values to customize the visibility of markup with MarkupVisibility[1200].

Possible values:

spyAuthenticMarkupHidden	= 0
spyAuthenticMarkupSmall	= 1
spyAuthenticMarkupLarge	= 2
spyAuthenticMarkupMixed	= 3

25.3.4.8 SPYAuthenticToolbarButtonState

Description
Authentic toolbar button states are given by the following enumeration:

Possible values:

authenticToolbarButtonDefault	= 0
authenticToolbarButtonEnabled	= 1
authenticToolbarButtonDisabled	= 2

25.3.4.9 SPYDatabaseKind

Description
Values to select different kinds of databases for import. See DatabaseConnection.DatabaseKind[1211] for its use.

(C) 2015-2021 Altova GmbH

Possible values:

spyDB_Access	= 0
spyDB_SQLServer	= 1
spyDB_Oracle	= 2
spyDB_Sybase	= 3
spyDB_MySQL	= 4
spyDB_DB2	= 5
spyDB_Other	= 6
spyDB_Unspecified	= 7
spyDB_PostgreSQL	= 8
spyDB_iSeries	= 9

25.3.4.10 SPYDialogAction

Description
Values to simulate different interactions on dialogs. See Dialogs 1215 for all dialogs available.

Possible values:

spyDialogOK	= 0	// simulate click on OK button
spyDialogCancel	= 1	// simulate click on Cancel button
spyDialogUserInput	= 2	// show dialog and allow user interaction

25.3.4.11 SPYDOMType

Description
Enumeration values to parameterize generation of C++ code from schema definitions.

Possible values:

spyDOMType_msxml4	= 0	Obsolete
spyDOMType_xerces	= 1	
spyDOMType_xerces3	= 2	
spyDOMType_msxml6	= 3	

`spyDOMType_xerces` indicates Xerces 2.x usage; `spyDOMType_xerces3` indicates Xerces 3.x usage.

25.3.4.12 SPYDTDSchemaFormat

Description
Enumeration to identify the different schema formats.

Possible values:

spyDTD	= 0
spyW3C	= 1

25.3.4.13 SPYEncodingByteOrder

Description
Enumeration values to specify encoding byte ordering for text import and export.

Possible values:

spyNONE	= 0
spyLITTLE_ENDIAN	= 1
spyBIG_ENDIAN	= 2

25.3.4.14 SPYExportNamespace

Description
Enumeration type to configure handling of namespace identifiers during export.

Possible values:

spyNoNamespace	= 0
spyReplaceColonWithUnderscore	= 1

25.3.4.15 SPYFindInFilesSearchLocation

Description
The different locations where a search can be performed. This type is used with the FindInFilesDlg 1261 dialog.

Possible values:

spyFindInFiles_Documents	= 0
spyFindInFiles_Project	= 1
spyFindInFiles_Folder	= 2

25.3.4.16 SPYFrequentElements

Description
Enumeration value to parameterize schema generation.

Possible values:

spyGlobalElements	= 0
spyGlobalComplexType	= 1

(C) 2015-2021 Altova GmbH

25.3.4.17 SPYImageKind

Description
Enumeration values to parameterize image type of the generated documentation. These values are used in SchemaDocumentationDialog.DiagramFormat 1284.

Possible values:

spyImageType_PNG	= 0
spyImageType_EMF	= 1

25.3.4.18 SPYImportColumnsType

Description
Enumeration to specify different Import columns types.

Possible values:

spyImportColumns_Element	= 0
spyImportColumns_Attribute	= 1

25.3.4.19 SPYKeyEvent

Description
Enumeration type to identify the different key events. These events correspond with the equally named windows messages.

Possible values:

spyKeyDown	= 0
spyKeyUp	= 1
spyKeyPressed	= 2

25.3.4.20 SPYKeyStatus

Description
Enumeration type to identify the key status.

Possible values:

spyLeftShiftKeyMask	= 1
spyRightShiftKeyMask	= 2
spyLeftCtrlKeyMask	= 4
spyRightCtrlKeyMask	= 8
spyLeftAltKeyMask	= 16
spyRightAltKeyMask	= 32

25.3.4.21 SPYLibType

Description
Enumeration values to parameterize generation of C++ code from schema definitions.

Possible values:

spyLibType_static	= 0
spyLibType_dll	= 1

25.3.4.22 SPYLoading

Description
Enumeration values to define loading behaviour of URL files.

Possible values:

spyUseCacheProxy	= 0
spyReload	= 1

25.3.4.23 SPYMouseEvent

Description
Enumeration type that defines the mouse status during a mouse event. Use the enumeration values as bitmasks rather then directly comparing with them.

Examples

```
' to check for ctrl-leftbutton-down in VB
If (i_eMouseEvent = (XMLSpyLib.spyLeftButtonDownMask Or XMLSpyLib.spyCtrlKeyDownMask)) Then
        ' react on ctrl-leftbutton-down
End If

' to check for double-click with any button in VBScript
If (((i_eMouseEvent And spyDoubleClickMask) <> 0) Then
        ' react on double-click
End If
```

Possible values:

spyNoButtonMask	= 0
spyMouseMoveMask	= 1
spyLeftButtonMask	= 2
spyMiddleButtonMask	= 4
spyRightButtonMask	= 8
spyButtonUpMask	= 16
spyButtonDownMask	= 32
spyDoubleClickMask	= 64
spyShiftKeyDownMask	= 128
spyCtrlKeyDownMask	= 256

(C) 2015-2021 Altova GmbH

spyLeftButtonDownMask	= 34	// spyLeftButtonMask | spyButtonDownMask
spyMiddleButtonDownMask	= 36	// spyMiddleButtonMask | spyButtonDownMask
spyRightButtonDownMask	= 40	// spyRightButtonMask | spyButtonDownMask
spyLeftButtonUpMask	= 18	// spyLeftButtonMask | spyButtonUpMask
spyMiddleButtonUpMask	= 20	// spyMiddleButtonMask | spyButtonUpMask
spyRightButtonUpMask	= 24	// spyRightButtonMask | spyButtonUpMask
spyLeftDoubleClickMask	= 66	// spyRightButtonMask | spyButtonUpMask
spyMiddleDoubleClickMask	= 68	// spyMiddleButtonMask | spyDoubleClickMask
spyRightDoubleClickMask	= 72	// spyRightButtonMask | spyDoubleClickMask

25.3.4.24 SPYNumberDateTimeFormat

Description
Enumeration value to configure database connections.

Possible values:

spySystemLocale	= 0
spySchemaCompatible	= 1

25.3.4.25 SPYProgrammingLanguage

Description
Enumeration values to select the programming language for code generation from schema definitions.

Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

Possible values:

spyUndefinedLanguage	= -1
spyJava	= 0
spyCpp	= 1
spyCSharp	= 2

25.3.4.26 SPYProjectItemTypes

Description
Enumeration values to identify the different elements in project item lists. See SpyProjectItem.ItemType 1298.

Possible values:

spyUnknownItem	= 0
spyFileItem	= 1
spyFolderItem	= 2
spyURLItem	= 3

25.3.4.27 SPYProjectType

Description
Enumeration values to parameterize generation of C# from schema definitions.

Possible values:

spyVisualStudioProject	= 0	Obsolete
spyVisualStudio2003Project	= 1	Obsolete
spyBorlandProject	= 2	Obsolete
spyMonoMakefile	= 3	Obsolete
spyVisualStudio2005Project	= 4	Obsolete
spyVisualStudio2008Project	= 5	Obsolete
spyVisualStudio2010Project	= 6	For C++ code also
spyVisualStudio2013Project	= 7	For C++ code also
spyVisualStudio2015Project	= 8	For C++ code also
spyVisualStudio2017Project	= 9	For C++ code also
spyVisualStudio2019Project	=10	For C++ code also

25.3.4.28 SpySampleXMLGenerationChoiceMode

Description
This enumeration is used in `GenerateSampleXMLDlg.ChoiceMode` 1274:

spySampleXMLGen_FirstBranch = 0
spySampleXMLGen_AllBranches = 1
spySampleXMLGen_ShortestBranch = 2

25.3.4.29 SPYSampleXMLGenerationOptimization (Obsolete)

This enumeration is OBSOLETE since v2014.

Description
Specify the elements that will be generated in the Sample XML.
This enumeration is used in GenerateSampleXMLDlg 1273.

Possible values:

spySampleXMLGen_Optimized	= 0
spySampleXMLGen_NonMandatoryElements	= 1
spySampleXMLGen_Everything	= 2

(C) 2015-2021 Altova GmbH

25.3.4.30 SpySampleXMLGenerationSampleValueHints

Description
This enumeration is used in GenerateSampleXMLDlg.SampleValueHints 1278

spySampleXMLGen_FirstFit = 0
spySampleXMLGen_RandomFit = 1
spySampleXMLGen_CycleThrough = 2

25.3.4.31 SPYSampleXMLGenerationSchemaOrDTDAssignment

Description
Specifies what kind of reference to the schema/DTD should be added to the generated Sample XML.
This enumeration is used in GenerateSampleXMLDlg 1273.

Possible values:

spySampleXMLGen_AssignRelatively = 0
spySampleXMLGen_AssignAbsolutely = 1
spySampleXMLGen_DoNotAssign = 2

25.3.4.32 SPYSchemaDefKind

Description
Enumeration type to select schema diagram types.

Possible values:

spyKindElement = 0
spyKindComplexType = 1
spyKindSimpleType = 2
spyKindGroup = 3
spyKindModel = 4
spyKindAny = 5
spyKindAttr = 6
spyKindAttrGroup = 7
spyKindAttrAny = 8
spyKindIdentityUnique = 9
spyKindIdentityKey = 10
spyKindIdentityKeyRef = 11
spyKindIdentitySelector = 12
spyKindIdentityField = 13

spyKindNotation = 14
spyKindInclude = 15
spyKindImport = 16
spyKindRedefine = 17
spyKindFacet = 18
spyKindSchema = 19
spyKindCount = 20

25.3.4.33 SPYSchemaDocumentationFormat

Description
Enumeration values to parameterize generation of schema documentation. These values are used in SchemaDocumentationDialog.OutputFormat 1290.

Possible values:

spySchemaDoc_HTML = 0
spySchemaDoc_MSWord = 1
spySchemaDoc_RTF = 2
spySchemaDoc_PDF = 3

25.3.4.34 SPYSchemaExtensionType

Description
Enumeration to specify different Schema Extension types.

Possible values:

spySchemaExtension_None = 0
spySchemaExtension_SQL_XML = 1
spySchemaExtension_MS_SQL_Server = 2
spySchemaExtension_Oracle = 3

25.3.4.35 SPYSchemaFormat

Description
Enumeration to specify different Schema Format types.

Possible values:

spySchemaFormat_Hierarchical = 0
spySchemaFormat_Flat = 1

(C) 2015-2021 Altova GmbH

25.3.4.36 SPYTextDelimiters

Description
Enumeration values to specify text delimiters for text export.

Possible values:

spyTabulator	= 0
spySemicolon	= 1
spyComma	= 2
spySpace	= 3

25.3.4.37 SPYTextEnclosing

Description
Enumeration value to specify text enclosing characters for text import and export.

Possible values:

spyNoEnclosing	= 0
spySingleQuote	= 1
spyDoubleQuote	= 2

25.3.4.38 SPYTypeDetection

Description
Enumeration to select how type detection works during `GenerateDTDOrSchema` 1233 and `GenerateDTDOrSchemaEx` 1233.

Possible values:

spyBestPossible	= 0
spyNumbersOnly	= 1
spyNoDetection	= 2

25.3.4.39 SPYURLTypes

Description
Enumeration to specify different URL types.

Possible values:

spyURLTypeAuto	= -1
spyURLTypeXML	= 0
spyURLTypeDTD	= 1

25.3.4.40 SPYValidateXSDVersion

Description
Enumeration values that select what XSD version to use. The XSD version that is selected depends on both (i) the presence/absence—and, if present, the value—of the `/xs:schema/@vc:minVersion` attribute of the XSD document, and (ii) the value of this enumeration.

`spyValidateXSDVersion_1_0` selects `XSD 1.0` if `vc:minVersion` is absent, or is present with any value.
`spyValidateXSDVersion_1_1` selects `XSD 1.1` if `vc:minVersion` is absent, or is present with any value.
`spyValidateXSDVersion_AutoDetect` selects `XSD 1.1` if `vc:minVersion=1.1`. If the `vc:minVersion` attribute is absent, or is present with a value other than `1.1`, then `XSD 1.0` is selected.

Possible values

```
spyValidateXSDVersion_AutoDetect      = 0
spyValidateXSDVersion_1_1             = 1
spyValidateXSDVersion_1_0             = 2
```

25.3.4.41 SPYValidateErrorFormat

Description
Enumeration values that select the format of the error message.

Possible values

```
spyValidateErrorFormat_Text           = 0
spyValidateErrorFormat_ShortXML       = 1
spyValidateErrorFormat_LongXML        = 2
```

25.3.4.42 SPYViewModes

Description
Enumeration values that define the different view modes for XML documents. The mode *spyViewAuthentic(4)* identifies the mode that was intermediately called DocEdit mode and is now called Authentic mode. The mode *spyViewJsonSchema* identifies a mode which is mapped to the Schema Design View on the GUI but is distinguished internally.

Possible values:

```
spyViewGrid          = 0
spyViewText          = 1
spyViewBrowser       = 2
spyViewSchema        = 3
spyViewContent       = 4     // obsolete
spyViewAuthentic     = 4
```

```
spyViewWSDL          = 5
spyViewZIP           = 6
spyViewEditionInfo   = 7
spyViewXBRL          = 8
spyViewJsonSchema    = 9
```

25.3.4.43 SPYVirtualKeyMask

Description
Enumeration type for the most frequently used key masks that identify the status of the virtual keys. Use these values as bitmasks rather then directly comparing with them. When necessary, you can create further masks by using the 'logical or' operator.

Examples

```
'  VBScript sample: check if ctrl-key is pressed
If ((i_nVirtualKeyStatus And spyCtrlKeyMask) <> 0)) Then
        ' ctrl-key is pressed
End If

'  VBScript sample: check if ONLY ctrl-key is pressed
If (i_nVirtualKeyStatus == spyCtrlKeyMask) Then
        ' exactly ctrl-key is pressed
End If

// JScript sample: check if any of the right virtual keys is pressed
if ((i_nVirtualKeyStatus & (spyRightShiftKeyMask | spyRightCtrlKeyMask | spyRightAltKeyMask)) != 0)
{
        ; ' right virtual key is pressed
}
```

Possible values:

```
spyNoVirtualKeyMask      = 0
spyLeftShiftKeyMask      = 1
spyRightShiftKeyMask     = 2
spyLeftCtrlKeyMask       = 4
spyRightCtrlKeyMask      = 8
spyLeftAltKeyMask        = 16
spyRightAltKeyMask       = 32
spyShiftKeyMask          = 3     // spyLeftShiftKeyMask | spyRightShiftKeyMask
spyCtrlKeyMask           = 12    // spyLeftCtrlKeyMask | spyRightCtrlKeyMask
spyAltKeyMask            = 48    // spyLeftAltKeyMask | spyRightAltKeyMask
```

25.3.4.44 SPYXMLDataKind

Description
The different types of XMLData elements available for XML documents.

Possible values:

spyXMLDataXMLDocStruct	= 0
spyXMLDataXMLEntityDocStruct	= 1
spyXMLDataDTDDocStruct	= 2
spyXMLDataXML	= 3
spyXMLDataElement	= 4
spyXMLDataAttr	= 5
spyXMLDataText	= 6
spyXMLDataCData	= 7
spyXMLDataComment	= 8
spyXMLDataPI	= 9
spyXMLDataDefDoctype	= 10
spyXMLDataDefExternalID	= 11
spyXMLDataDefElement	= 12
spyXMLDataDefAttlist	= 13
spyXMLDataDefEntity	= 14
spyXMLDataDefNotation	= 15
spyXMLDataKindsCount	= 16

25.3.5 Application API for Java (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java [1118].

The Application API in Java has an interface built up of Java classes, each of which corresponds to an object in the Application API [1095]. Developers can use these Java classes to interact with the COM API. These classes are listed below and described in subsequent sections. For a description of the Application API [1095] objects themselves, see the Application API documentation [1095]. Bear in mind that some API features are only available in scripting environments; these have therefore not been ported to Java.

Java classes

- SpyApplication [1375]
  - SpyProject [1387]
    - SpyProjectItems [1388]
      - SpyProjectItem [1388]
  - SpyDocuments [1380]
    - SpyDoc [1378]
      - SpyAuthenticView [1394]
        - SpyAuthenticRange [1392]
      - SpyDocEditView [1395]
        - SpyDocEditSelection [1395]
      - SpyGridView [1387]
      - SpyTextView [1391]
      - SpyXMLData [1392]
  - SpyDialogs [1378]
    - SpyCodeGeneratorDlg [1376]
    - SpyDTDSchemaGeneratorDlg [1381]

SpyFileSelectionDlg 1383
SpyFindInFilesDlg 1383
SpyGenerateSampleXMLDlg 1386
SpySchemaDocumentationDlg 1389
SpyDatabaseConnection 1377
SpyElementList 1381
SpyElementListItem 1382
SpyExportSettings 1382
SpyFindInFilesResults 1385
SpyFindInFilesResult 1385
SpyFindInFilesMatch 1385
SpyTextImportExportSettings 1390

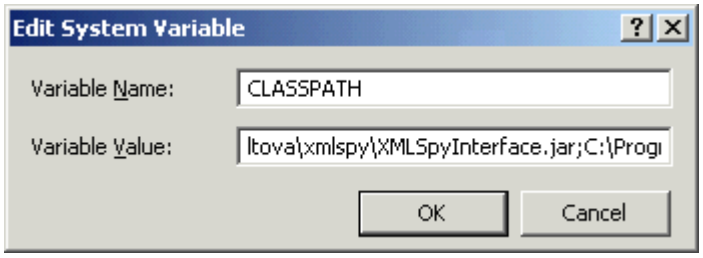
Implementation of COM properties in Java
Properties in Java have been defined to include both a **set** and **get** method (set if it is allowed by the COM implementation). For example, the COM class Document contains the **GridView** property. In Java the method is called SpyDoc and the property is defined as a **GetGridView** method.

If you encounter compiling problems, please check the following points:

- The xmlspylib.dll must be available in ..¥windows¥system32.
- The XMLSpyInterface.jar file must be inserted in the ClassPath environment variable.

Setting the ClassPath variable in Windows XP

1. Click **Start | Settings | Control panel | System | Advanced | Environment Variables**. This opens the Environment Variables dialog box.
2. If a ClassPath entry already exists in the **System variables** group, select the ClassPath entry, and click the **Edit** button. Edit the path to: "C:¥Program Files¥Altova¥xmlspy¥XMLSpyInterface.jar".



If a ClassPath entry does not exist in the System variables group, click the **New** button. The New System Variable dialog pops up. Enter CLASSPATH as the variable name, and "C:¥Program Files¥Altova¥xmlspy¥XMLSpyInterface.jar" as the ClassPath variable (alter the path to match your installation, if necessary).

25.3.5.1 Sample source code (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java 1118.

The "`SpyDoc doc = app.GetDocuments().OpenFile(...)`" command parameter must be altered to suit your environment.

What the sample does:

- Starts a new XMLSpy instance
- Opens the Datasheet.xml file (alter the path here...)
- Switches to the Enhanced Grid view
- Appends a new child element called "NewChild" with the text value "NewValuE" element to the root element
- Checks if the document is valid and outputs a message to the Java console
- Quits and releases the XMLSpy application

```
import XMLSpyInterface.*;

public class TestSpyInterface
{
    public TestSpyInterface() {}

    public static void main(String[] args)
    {
        SpyApplication app = null;
        SpyDoc oDoc = null;
        SpyXMLData oData = null;
        SpyXMLData oNewChild = null;

        try
        {
            app = new SpyApplication();
            app.ShowApplication( true );

            oDoc = app.GetDocuments().OpenFile("C:¥¥FilePath¥¥OrgChart.xml", true );

            // OrgChart.xml is in the folder C:¥Documents and Settings¥<ユーザー名>¥マイドキュメント
            ¥Altova¥XMLSpy2021. The filepath should be in
            // the form: C:¥¥Documents and Settings¥¥Username¥¥Folder¥¥Filename.xml

            if ( oDoc != null )
            {
                oDoc.SwitchViewMode(SPYViewModes.spyViewGrid);
                oData = oDoc.GetRootElement();
                oNewChild = oDoc.CreateChild(SPYXMLDataKind.spyXMLDataElement);

                oNewChild.SetName( "NewChild" );
                oNewChild.SetTextValue("newVaLuE");
                oData.AppendChild(oNewChild);

                if ( oDoc.IsValid() == false )
                {
                    // is to be expected after above insertion
                    System.out.println( "!!!!!!validation error: " + oDoc.GetErrorString() );
                    System.out.println( "!!!!!!validation error: " + oDoc.GetErrorPos() );
                    System.out.println( "!!!!!!validation error: " + oDoc.GetBadData() );
                }
            }

            app.Quit();
```

```
        }
        finally
        {
            // Free any allocated resources by calling ReleaseInstance().
            if ( oNewChild != null )
                oNewChild.ReleaseInstance();

            if ( oData != null )
                oData.ReleaseInstance();

            if ( oDoc != null )
                oDoc.ReleaseInstance();

            if ( app != null )
                app.ReleaseInstance();
        }
    }
}
```

If you have difficulties compiling this sample, please try the following commands on the (**Start | Run | cmd**) command line. Please make sure you are currently in the folder that contains the sample java file.

compilation
javac -classpath c:*yourpathhere*\XMLSpyInterface.jar *testspyinterface.java*

Execution
java -classpath c:*yourpathhere*\XMLSpyInterface.jar *testspyinterface*

25.3.5.2 SpyApplication (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java [1118].

```
public class SpyApplication
{
  public void ReleaseInstance[1373]();
  public void ShowApplication[1148]( boolean bShow );
  public void Quit[1146]();
  public void AddMacroMenuItem[1134]( String sMacro, String sDisplayText );
  public void ClearMacroMenu[1135]();
  public SpyDoc[1378] GetActiveDocument[1134]();
  public SpyProject[1387] GetCurrentProject[1136]();
  public SpyDocuments[1380] GetDocuments[1136]();
  public SpyElementList[1381] GetDatabaseImportElementList[1137]( SpyDatabaseConnection[1377] oImportSettings );
  public SpyDatabaseConnection[1377] GetDatabaseSettings[1138]();
  public SpyElementList[1381] GetDatabaseTables[1138]( SpyDatabaseConnection[1377] oImportSettings );
  public SpyExportSettings[1382] GetExportSettings[1139]();
  public SpyElementList[1381] GetTextImportElementList[1139]( SpyTextImportExportSettings[1390] oImportSettings );
  public SpyTextImportExportSettings[1390] GetTextImportExportSettings[1140]();
  public SpyDoc[1378] ImportFromDatabase[1141]( SpyDatabaseConnection[1377] oImportSettings, SpyElementList[1381]
  oElementList );
  public SpyDoc[1378] ImportFromSchema[1142]( SpyDatabaseConnection[1377] oImportSettings, String strTable, SpyDoc[1378]
  oSchemaDoc );
```

```
   public SpyDoc 1378 ImportFromText 1143( SpyTextImportExportSettings 1390 oImportSettings, SpyElementList 1381
   oElementList );
   public SpyDoc 1378 ImportFromWord 1144( String sFile );
   public void NewProject 1145( String sPath, boolean bDiscardCurrent );
   public void OpenProject 1145(String sPath , boolean bDiscardCurrent, boolean bDialog );
   public long ShowForm 1148( String sName );
   public void URLDelete 1149( String sURL, String sUser, String sPassword );
   public void URLMakeDirectory 1149( String sURL, String sUser, String sPassword );
   public int GetWarningNumber 1150();
   public String GetWarningText 1150();

   // since Version 2004R4
   public SpyApplication 1375 GetApplication 1135();
   public SpyApplication 1375 GetParent 1145();
   public SpyDialogs 1378 GetDialogs 1136();
   public boolean GetVisible 1149();
   public void SetVisible 1149( boolean i_bVisibility );
   public long GetWindowHandle();

   public void ReloadSettings 1146();
   public SpyFindInFilesResults 1385 FindInFiles 1137( SpyFindInFilesDlg 1383 dlgSettings );
   public boolean ShowFindInFiles 1148( SpyFindInFilesDlg 1383 dlgSettings );
   public void Selection( String sVal );

   public long 1396 Status 1149();
   public int MajorVersion 1144();
   public int MinorVersion 1144();
   public String Edition 1137();
   public boolean IsAPISupported 1144();
   public long ServicePackVersion 1147();
   public void CreateXMLSchemaFromDBStructure 1135( SpyDatabaseConnection 1377 oConnection, SpyElementList 1381
oTables );
}
```

25.3.5.3 SpyCodeGeneratorDlg (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java 1118.

Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

```
// since version 2004R4
public class SpyCodeGeneratorDlg
{
   public void ReleaseInstance 1373();
   public SpyApplication 1375 GetApplication 1204();
   public SpyDialogs 1378 GetParent 1208();
   public long 1404 GetProgrammingLanguage 1208();
   public void SetProgrammingLanguage 1208( long 1404 i_eVal );
   public String GetTemplateFileName 1209();
   public void SetTemplateFileName 1209( String i_strVal );
   public String GetOutputPath 1207();
   public void SetOutputPath 1207( String i_strVal );
   public long 1400 GetOutputPathDialogAction 1207();
```

```
    public void SetOutputPathDialogAction( long i_eVal );
    public long GetPropertySheetDialogAction();
    public void SetPropertySheetDialogAction( long i_eVal );
    public long GetOutputResultDialogAction();
    public void SetOutputResultDialogAction( long i_eVal );
    public long GetCPPSettings_DOMType();
    public void SetCPPSettings_DOMType( long i_eVal );
    public long GetCPPSettings_LibraryType();
    public void SetCPPSettings_LibraryType( long i_eVal );
    public boolean GetCPPSettings_UseMFC();
    public void SetCPPSettings_UseMFC( boolean i_bVal );
    public long GetCSharpSettings_ProjectType();
    public void SetCSharpSettings_ProjectType( long i_eVal );
}
```

25.3.5.4 SpyDatabaseConnection (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java .

```
public class SpyDatabaseConnection
{
    public void ReleaseInstance();
    public String GetADOConnection();
    public void SetADOConnection( String sValue );
    public boolean GetAsAttributes();
    public void SetAsAttributes( boolean bValue );
    public boolean GetCreateMissingTables();
    public void SetCreateMissingTables( boolean bValue );
    public boolean GetCreateNew();
    public void SetCreateNew( boolean bValue );
    public boolean GetExcludeKeys();
    public void SetExcludeKeys( boolean bValue );
    public String GetFile();
    public void SetFile( String sValue );
    public boolean GetIncludeEmptyElements();
    public void SetIncludeEmptyElements( boolean bValue );
    public long GetNumberDateTimeFormat();
    public void SetNumberDateTimeFormat( long nValue );
    public String GetODBCConnection();
    public void SetODBCConnection( String sValue );
    public String GetSQLSelect();
    public void SetSQLSelect( String sValue );
    public long GetTextFieldLen();
    public void SetTextFieldLen( long nValue );

    // since version 2004R4
    public long GetDatabaseKind();
    public void SetDatabaseKind( long nValue );
```

```
   // since version 2008R2
   public boolean GetCommentIncluded[1210]();
   public void SetCommentIncluded[1210]( boolean bValue );
   public String GetNullReplacement[1213]();
   public void SetNullReplacement[1213]( String sValue );
   public String GetDatabaseSchema[1211]();
   public void SetDatabaseSchema[1211]( String sValue );

   // since version 2010r3
   public boolean GetPrimaryKeys[1214]()
   public void SetPrimaryKeys[1214]( boolean bValue )
   public boolean GetForeignKeys[1212]()
   public void SetForeignKeys[1212]( boolean bValue )
   public boolean GetUniqueKeys[1215]()
   public void SetUniqueKeys[1215]( boolean bValue )
   public long[1408] GetSchemaExtensionType[1214]()
   public void SetSchemaExtensionType[1214]( long[1408] nValue )
   public long[1408] GetSchemaFormat[1214]()
   public void SetSchemaFormat[1214]( long[1408] nValue )
   public long[1403] GetImportColumnsType[1212]()
   public void SetImportColumnsType[1212]( long[1403] nValue )
}
```

25.3.5.5 SpyDialogs (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java[1118].

```
// Since version 2004R4
public class SpyDialogs
{
   public SpyApplication[1375] GetApplication[1215]();
   public SpyApplication[1375] GetParent[1216]();
   public SpyCodeGeneratorDlg[1376] GetCodeGeneratorDlg[1216]();
   public SpyFileSelectionDlg[1383] GetFileSelectionDlg[1216]();
   public SpySchemaDocumentationDlg[1389] GetSchemaDocumentationDlg[1217]();
   public SpyGenerateSampleXMLDlg[1386] GetGenerateSampleXMLDlg[1217]();
   public SpyDTDSchemaGeneratorDlg[1381] GetDTDSchemaGeneratorDlg[1217]();
   public SpyFindInFilesDlg[1383] GetFindInFilesDlg[1217]();
}
```

25.3.5.6 SpyDoc (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

(C) 2015-2021 Altova GmbH

For information about how to access the Application API from Java code, see the section: Programming Languages | Java [1118].

```
public class SpyDoc
{
   public void ReleaseInstance [1373]();
   public void SetEncoding [1229]( String strEncoding );
   public void SetPathName [1229]( String strPath );
   public String GetPathName [1236]();
   public String GetTitle [1245]();
   public boolean IsModified [1236]();
   public void Save [1240]();
   public void Close [1225]( boolean bDiscardChanges );
   public void UpdateViews [1246]();
   public long [1410] GetCurrentViewMode [1228]();
   public boolean SwitchViewMode [1244]( long [1410] nMode );
   public SpyGridView [1387] GetGridView [1236]();
   public void SetActiveDocument [1242]();
   public void StartChanges [1244]();
   public void EndChanges [1229]();
   public void TransformXSL [1245]();
   public void AssignDTD [1222]( String sDTDFile, boolean bDialog );
   public void AssignSchema [1223]( String sSchemaFile, boolean bDialog );
   public void AssignXSL [1223]( String sXSLFile, boolean bDialog );
   public void ConvertDTDOrSchema [1225]( long [1400] nFormat, long [1402] nFrequentElements );
   public SpyXMLData [1392] CreateChild [1227]( long [1411] nKind );
   public void CreateSchemaDiagram [1227]( long [1406] nKind, String sName, String sFile );
   public SpyDocEditView [1395] GetDocEditView [1228]();
   public void ExportToDatabase [1230]( SpyXMLData [1392] oFromChild, SpyExportSettings [1382] oExportSettings,
   SpyDatabaseConnection [1377] oDatabaseConnection );
   public void ExportToText [1231]( SpyXMLData [1392] oFromChild, SpyExportSettings [1382] oExportSettings,
   SpyTextImportExportSettings [1390] oTextSettings );
   public void GenerateDTDOrSchema [1233]( long [1400] nFormat, int nValuesList, long [1409] nDetection, long [1402]
   nFrequentElements );
   public SpyElementList [1381] GetExportElementList [1235]( SpyXMLData [1392] oFromChild, SpyExportSettings [1382]
   oExportSettings );
   public SpyXMLData [1392] GetRootElement [1240]();
   public String SaveInString [1241]( SpyXMLData [1392] oData, boolean bMarked );
   public void SaveToURL [1242]( String sUrl, String sUser, String sPassword );
   public String GetErrorString(); // See IsValid [1236]() or IsWellFormed [1239]()
   public int GetErrorPos(); // See IsValid [1236]() or IsWellFormed [1239]()
   public SpyXMLData [1392] GetBadData(); // See IsValid [1236]() or IsWellFormed [1239]()
   public boolean IsValid [1236]();
   public boolean IsWellFormed [1239]( SpyXMLData [1392] oData, boolean bWithChildren );

   // Since version 2004R3
   public SpyAuthenticView [1394] GetAuthenticView [1224]()

   // Since version 2004R4
   public SpyApplication [1375] GetApplication [1222]();
   public SpyDocuments [1380] GetParent [1239]();
   public String GetFullName [1232]();
   public void SetFullName [1232]( String i_strName );
   public String GetName [1239]();
   public String GetPath [1240]();
   public boolean GetSaved [1241]();
   public void SaveAs [1241]( String i_strFileNameOrPath );
   public String GetEncoding [1229]();
   public SpyXMLData [1392] GetDataRoot [1228]();
```

```
   public void GenerateProgramCode[1234]( SpyCodeGeneratorDlg[1376] i_dlg );
   public void AssignXSLFO[1223]( String i_strFile, boolean i_bUseDialog );
   public void TransformXSLFO[1245]();
   public void GenerateSchemaDocumentation[1234]( SpySchemaDocumentationDlg[1389] i_dlg );

   public void ExecuteXQuery[1229]( String i_strXMLSourceFile );
   public void SetExternalIsValid[1243]( boolean bIsValid );
   public SpyDoc[1378] GenerateSampleXML[1234]( SpyGenerateSampleXMLDlg[1386] ipGenerateXMLDlg );
   public boolean UpdateXMLData[1246]();
   public String GetAsXMLString[1224]();
   public void SetAsXMLString[1224]( String newVal );
   public SpyDoc[1378] GenerateDTDOrSchemaEx[1233]( SpyDTDSchemaGeneratorDlg[1381] ipDTDSchemaGeneratorDlg );
   public SpyDoc[1378] ConvertDTDOrSchemaEx[1225]( long[1400] nFormat, long[1402] nFrequentElements, String sOutputPath,
   long[1400] nOutputPathDialogAction );
   public SpyTextView[1391] GetTextView[1244]();
   public String[] GetSuggestions[1244]();
   public void SetSuggestions[1244]( String[] aList );
   public void SetSelection( String sVal );

   // Since version 2009
   public void TransformXSLEx[1245]( long[1400] nDialogAction );

   // Since version 2010r3
   public String CreateDBStructureFromXMLSchema[1227]( SpyDatabaseConnection[1377] oConnection, SpyElementList[1381]
   oTables, boolean bDropTableWithExistingName );
   public SpyElementList[1381] GetDBStructureList[1235]( SpyDatabaseConnection[1377] oConnection );
}
```

25.3.5.7 SpyDocuments (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java[1118].

```
public class SpyDocuments
{
   public void ReleaseInstance[1373]();
   public long Count[1247]();
   public SpyDoc[1378] GetItem[1248]( long nNo );
   public SpyDoc[1378] NewFile[1248]( String strFile, String strType );
   public SpyDoc[1378] NewFileFromText[1249]( String nSource, String strType );
   public SpyDoc[1378] OpenFile[1249]( String sPath, boolean bDialog );
   public SpyDoc[1378] OpenURL[1250]( String sUrl, long[1410] nURLType, long[1404] nLoading, String sUser, String sPassword );
   public SpyDoc[1378] OpenURLDialog[1250](String sURL, long[1410] nURLType, long[1404] nLoading, String sUser, String
   sPassword );
   // Since version 2011r2
   public SpyDoc[1378] NewAuthenticFile[1248]( String strSPSPath, String strXMLPath );
   public SpyDoc[1378] OpenAuthenticFile[1249]( String strSPSPath, String strXMLPath );
}
```

25.3.5.8 SpyDTDSchemaGeneratorDlg (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java 1118.

```
public class SpyDTDSchemaGeneratorDlg
{
  public void ReleaseInstance 1373 ();
  public SpyApplication 1375 GetApplication 1251 ();
  public long 1400 GetDTDSchemaFormat 1252 ();
  public void SetDTDSchemaFormat 1252 ( long 1400 newVal );
  public short GetValueList 1255 ();
  public void SetValueList 1255 ( short newVal );
  public long 1409 GetTypeDetection 1255 ();
  public void SetTypeDetection 1255 ( long 1409 newVal );
  public long 1402 GetFrequentElements 1252 ();
  public void SetFrequentElements 1252 ( long 1402 newVal );
  public boolean GetMergeAllEqualNamed 1253 ();
  public void SetMergeAllEqualNamed 1253 ( boolean newVal );
  public boolean GetResolveEntities 1254 ();
  public void SetResolveEntities 1254 ( boolean newVal );
  public long 1397 GetAttributeTypeDefinition 1252 ();
  public void SetAttributeTypeDefinition 1252 ( long 1397 newVal );
  public boolean GetGlobalAttributes 1252 ();
  public void SetGlobalAttributes 1252 ( boolean newVal );
  public boolean GetOnlyStringEnums 1253 ();
  public void SetOnlyStringEnums 1253 ( boolean newVal );
  public long GetMaxEnumLength 1253 ();
  public void SetMaxEnumLength 1253 ( long newVal );
  public String GetOutputPath 1253 ();
  public void SetOutputPath 1253 ( String newVal );
  public long 1400 GetOutputPathDialogAction 1254 ();
  public void SetOutputPathDialogAction 1254 ( long 1400 newVal );
}
```

25.3.5.9 SpyElementList (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java [1118].

```
public class SpyElementList
{
   public void ReleaseInstance [1373]();
   public long GetCount [1256]();
   public SpyElementListItem [1382] GetItem [1256]( long nIndex );
   public void RemoveElement [1256]( long nIndex );
}
```

25.3.5.10 SpyElementListItem (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java [1118].

```
public class SpyElementListItem
{
   public void ReleaseInstance [1373]();
   public long [1411] GetElementKind [1256]();
   public void SetElementKind [1256]( long [1411] nKind );
   public long GetFieldCount [1257]();
   public String GetName [1257]();
   public long GetRecordCount [1257]();
}
```

25.3.5.11 SpyExportSettings (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java [1118].

```
public class SpyExportSettings
{
   public void ReleaseInstance [1373]();
   public boolean GetCreateKeys [1258]();
   public void SetCreateKeys [1258]( boolean bValue );
   public SpyElementList [1381] GetElementList [1258]();
   public void SetElementList [1258]( SpyElementList [1381] obj );
   public boolean GetEntitiesToText [1258] ();
```


(C) 2015-2021 Altova GmbH

```
  public void SetEntitiesToText[1258]( boolean bValue );
  public boolean GetExportAllElements[1258]();
  public void SetExportAllElements[1258]( boolean bValue );
  public boolean GetFromAttributes[1259]();
  public void SetFromAttributes[1259]( boolean bValue );
  public boolean GetFromSingleSubElements[1259]();
  public void SetFromSingleSubElements[1259]( boolean bValue );
  public boolean GetFromTextValues[1259]();
  public void SetFromTextValues[1259]( boolean bValue );
  public boolean GetIndependentPrimaryKey[1259]();
  public void SetIndependentPrimaryKey[1259]( boolean bValue );
  public long[1401] GetNamespace[1259]();
  public void SetNamespace[1259]( long[1401] nValue );
  public int GetSubLevelLimit[1260]();
  public void SetSubLevelLimit[1260]( int nValue );
}
```

25.3.5.12 SpyFileSelectionDlg (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java[1118].

```
// Since version 2004R4
public class SpyFileSelectionDlg
{
  public void ReleaseInstance[1373]();
  public SpyApplication[1375] GetApplication[1260]();
  public SpyDialogs[1378] GetParent[1261]();
  public String GetFullName[1261]();
  public void SetFullName[1261]( String i_strName );
  public long[1400] GetDialogAction[1260]();
  public void SetDialogAction[1260]( long[1400] i_eAction );
}
```

25.3.5.13 SpyFindInFilesDlg (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java[1118].

```
public class SpyFindInFilesDlg
{
  public void ReleaseInstance[1373]();
```

```
  public SpyApplication[1375] GetApplication[1262]();
  public String GetFind[1263]();
  public void SetFind[1263]( String sNewVal );
  public boolean GetRegularExpression[1264]();
  public void SetRegularExpression[1264]( boolean bNewVal );
  public String GetReplace[1265]();
  public void SetReplace[1265]( String sNewVal );
  public boolean GetReplaceOnDisk[1265]();
  public void SetReplaceOnDisk[1265]( boolean bNewVal );
  public boolean GetDoReplace[1263]();
  public void SetDoReplace[1263]( boolean bNewVal );
  public boolean GetMatchWholeWord[1264]();
  public void SetMatchWholeWord[1264]( boolean bNewVal );
  public boolean GetMatchCase[1264]();
  public void SetMatchCase[1264]( boolean bNewVal );
  public long[1402] GetSearchLocation[1265]();
  public void SetSearchLocation[1265]( long[1402] nPosition );
  public String GetStartFolder[1266]();
  public void SetStartFolder[1266]( String sNewVal );
  public boolean GetIncludeSubfolders[1263]();
  public void SetIncludeSubfolders[1263]( boolean bNewVal );
  public boolean GetSearchInProjectFilesDoExternal[1265]();
  public void SetSearchInProjectFilesDoExternal[1265]( boolean bNewVal );
  public String GetFileExtension[1263]();
  public void SetFileExtension[1263]( String sNewVal );
  public boolean GetAdvancedXMLSearch[1262]();
  public void SetAdvancedXMLSearch[1262]( boolean bNewVal );
  public boolean GetXMLElementNames[1267]();
  public void SetXMLElementNames[1267]( boolean bNewVal );
  public boolean GetXMLElementContents[1267]();
  public void SetXMLElementContents[1267]( boolean bNewVal );
  public boolean GetXMLAttributeNames[1266]();
  public void SetXMLAttributeNames[1266]( boolean bNewVal );
  public boolean GetXMLAttributeContents[1266]();
  public void SetXMLAttributeContents[1266]( boolean bNewVal );
  public boolean GetXMLComments[1267]();
  public void SetXMLComments[1267]( boolean bNewVal );
  public boolean GetXMLCData[1267]();
  public void SetXMLCData[1267]( boolean bNewVal );
  public boolean GetXMLPI[1268]();
  public void SetXMLPI[1268]( boolean bNewVal );
  public boolean GetXMLRest[1268]();
  public void SetXMLRest[1268]( boolean bNewVal );
  public boolean GetShowResult[1266]();
  public void SetShowResult[1266]( boolean bNewVal );
}
```

25.3.5.14 SpyFindInFilesMatch (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java 1118.

```
public class SpyFindInFilesMatch
{
  public void ReleaseInstance 1373 ();
  public long Line 1271 ();
  public long Position 1272 ();
  public long Length 1270 ();
  public String LineText 1271 ();
  public boolean Replaced 1272 ();
}
```

25.3.5.15 SpyFindInFilesResult (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java 1118.

```
public class SpyFindInFilesResult
{
  public void ReleaseInstance 1373 ();
  public long Count 1269 ();
  public SpyFindInFilesMatch 1385 GetItem 1269 ( long nNo );
  public String GetPath 1270 ();
  public SpyDoc 1378 GetDocument 1269 ();
}
```

25.3.5.16 SpyFindInFilesResults (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java[1118].

```
public class SpyFindInFilesResults
{
  public void ReleaseInstance[1373]();
  public long Count[1273]();
  public SpyFindInFilesResult[1385] GetItem[1273]( long nNo );
}
```

25.3.5.17 SpyGenerateSampleXMLDlg (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java[1118].

```
public class SpyGenerateSampleXMLDlg
{
  public void ReleaseInstance[1373]();
  public SpyApplication[1375] GetApplication[1274]();
  public boolean GetNonMandatoryAttributes[1276]();
  public void SetNonMandatoryAttributes[1276]( boolean newVal );
  public boolean GetNonMandatoryElements[1276]();
  public void SetNonMandatoryElements[1276]( boolean newVal );
  public boolean GetTakeFirstChoice[1278]();
  public void SetTakeFirstChoice[1278]( boolean newVal );
  public long GetRepeatCount[1277]();
  public void SetRepeatCount[1277]( long newVal );
  public boolean GetFillWithSampleData[1275]();
  public void SetFillWithSampleData[1275]( boolean newVal );
  public boolean GetFillElementsWithSampleData[1275]();
  public void SetFillElementsWithSampleData[1275]( boolean newVal );
  public boolean GetFillAttributesWithSampleData[1275]();
  public void SetFillAttributesWithSampleData[1275]( boolean newVal );
  public boolean GetContentOfNillableElementsIsNonMandatory[1275]();
  public void SetContentOfNillableElementsIsNonMandatory[1275]( boolean newVal );
  public boolean GetTryToUseNonAbstractTypes[1278]();
  public void SetTryToUseNonAbstractTypes[1278]( boolean newVal );
  public long[1406] GetOptimization[1277]();
  public void SetOptimization[1277]( long[1406] newVal );
  public long[1406] GetSchemaOrDTDAssignment[1278]();
  public void SetSchemaOrDTDAssignment[1278]( long[1406] newVal );
  public String GetLocalNameOfRootElement[1276]();
  public void SetLocalNameOfRootElement[1276]( String newVal );
```


(C) 2015-2021 Altova GmbH

```
  public String GetNamespaceURIOfRootElement();
  public void SetNamespaceURIOfRootElement( String newVal );
  public long GetOptionsDialogAction();
  public void SetOptionsDialogAction( long newVal );
}
```

25.3.5.18 SpyGridView (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java.

```
public class SpyGridView
{
  public void ReleaseInstance();
  public SpyXMLData GetCurrentFocus();
  public void Deselect( SpyXMLData oData );
  public boolean GetIsVisible();
  public void Select( SpyXMLData oData );
  public void SetFocus( SpyXMLData oData );
}
```

25.3.5.19 SpyProject (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java.

```
public class SpyProject
{
  public void ReleaseInstance();
  public void CloseProject( boolean bDiscardChanges, boolean bCloseFiles, boolean bDialog );
  public String GetProjectFile();
  public void SetProjectFile( String sFile );
  public SpyProjectItems GetRootItems();
  public void SaveProject();
  public void SaveProjectAs( String sPath, boolean bDialog );
}
```

25.3.5.20 SpyProjectItem (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java [1118].

```
public class SpyProjectItem
{
   public void ReleaseInstance[1373]();
   public SpyProjectItems[1388] GetChildItems[1298]();
   public String GetFileExtensions[1298]();
   public void SetFileExtensions[1298]( String sExtensions );
   public long[1405] GetItemType[1298]();
   public String GetName[1298]();
   public SpyDoc[1373] Open[1298]();
   public SpyProjectItem[1388] GetParentItem[1298]();
   public String GetPath[1299]();
   public String GetValidateWith[1299]();
   public void SetValidateWith[1299]( String sVal );
   public String GetXMLForXSLTransformation[1299]();
   public void SetXMLForXSLTransformation[1299]( String sVal );
   public String GetXSLForXMLTransformation[1299]();
   public void SetXSLForXMLTransformation[1299]( String sVal );
   public String GetXSLTransformationFileExtension[1299]();
   public void SetXSLTransformationFileExtension[1299]( String sVal );
   public String GetXSLTransformationFolder[1299]();
   public void SetXSLTransformationFolder[1299]( String sVal );
}
```

25.3.5.21 SpyProjectItems (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java [1118].

```
public class SpyProjectItems
{
   public void ReleaseInstance[1373]();
   public void AddFile[1300]( String sPath );
   public void AddFolder[1300]( String sName );
   public void AddURL[1300]( String sURL, long[1410] nURLType, String sUser, String sPassword, boolean bSave );
   public long Count[1301]();
   public SpyProjectItem[1388] GetItem[1301]( long nNumber );
   public void RemoveItem[1301]( SpyProjectItem[1388] oItemToRemove );
}
```


(C) 2015-2021 Altova GmbH

25.3.5.22 SpySchemaDocumentationDlg (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java [1118].

```
// Since version 2004R4
public class SpySchemaDocumentationDlg
{
    public void ReleaseInstance[1373]();
    public SpyApplication[1375] GetApplication[1284]();
    public SpyDialogs[1378] GetParent[1290]();

    public String GetOutputFile[1290]();
    public void SetOutputFile[1290]( String i_strVal );
    public long[1407] GetOutputFormat[1290]();
    public void SetOutputFormat[1290]( long[1407] i_eVal );

    public boolean GetShowResult[1294]();
    public void SetShowResult[1294]( boolean i_bVal );
    public long[1400] GetOptionsDialogAction[1289]();
    public void SetOptionsDialogAction[1289]( long[1400] i_eVal );
    public long[1400] GetOutputFileDialogAction[1290]();
    public void SetOutputFileDialogAction[1290]( long[1400] i_eVal );
    public boolean GetShowProgressBar[1293]();
    public void SetShowProgressBar[1293]( boolean i_bVal );

    public void IncludeAll[1285]( boolean i_bInclude );
    public boolean GetIncludeIndex[1287]();
    public void SetIncludeIndex[1287]( boolean i_bVal );
    public boolean GetIncludeGlobalElements[1287]();
    public void SetIncludeGlobalElements[1287]( boolean i_bVal );
    public boolean GetIncludeLocalElements[1288]();
    public void SetIncludeLocalElements[1288]( boolean i_bVal );
    public boolean GetIncludeGroups[1287]();
    public void SetIncludeGroups[1287]( boolean i_bVal );
    public boolean GetIncludeComplexTypes[1286]();
    public void SetIncludeComplexTypes[1286]( boolean i_bVal );
    public boolean GetIncludeSimpleTypes[1289]();
    public void SetIncludeSimpleTypes[1289]( boolean i_bVal );
    public boolean GetIncludeAttributeGroups[1286]();
    public void SetIncludeAttributeGroups[1286]( boolean i_bVal );
    public boolean GetIncludeRedefines[1288]();
    public void SetIncludeRedefines[1288]( boolean i_bVal );

    public void AllDetails[1283]( boolean i_bDetailsOn );
    public boolean GetShowDiagram[1292]();
    public void SetShowDiagram[1292]( boolean i_bVal );
    public boolean GetShowNamespace[1293]();
    public void SetShowNamespace[1293]( boolean i_bVal );
    public boolean GetShowType[1295]();
    public void SetShowType[1295]( boolean i_bVal );
    public boolean GetShowChildren[1291]();
```

```
public void SetShowChildren 1291( boolean i_bVal );
public boolean GetShowUsedBy 1295();
public void SetShowUsedBy 1295( boolean i_bVal );
public boolean GetShowProperties 1293();
public void SetShowProperties 1293( boolean i_bVal );
public boolean GetShowSingleFacets 1294();
public void SetShowSingleFacets 1294( boolean i_bVal );
public boolean GetShowPatterns 1293();
public void SetShowPatterns 1293( boolean i_bVal );
public boolean GetShowEnumerations 1292();
public void SetShowEnumerations 1292( boolean i_bVal );
public boolean GetShowAttributes 1291();
public void SetShowAttributes 1291( boolean i_bVal );
public boolean GetShowIdentityConstraints 1292();
public void SetShowIdentityConstraints 1292( boolean i_bVal );
public boolean GetShowAnnotations 1291();
public void SetShowAnnotations 1291( boolean i_bVal );
public boolean GetShowSourceCode 1294();
public void SetShowSourceCode 1294( boolean i_bVal );

// Since version 2009
public boolean GetEmbedDiagrams 1285();
public void SetEmbedDiagrams 1285( boolean i_bVal );
public long 1402 GetDiagramFormat 1284();
public void SetDiagramFormat 1284( long 1402 i_nVal );
public boolean GetIncludeGlobalAttributes 1286();
public void SetIncludeGlobalAttributes 1286( boolean i_bVal );
public boolean GetIncludeLocalAttributes 1287();
public void SetIncludeLocalAttributes 1287( boolean i_bVal );
public boolean GetIncludeReferencedSchemas 1288();
public void SetIncludeReferencedSchemas 1288( boolean i_bVal );
public boolean GetMultipleOutputFiles 1289();
public void SetMultipleOutputFiles 1289( boolean i_bVal );

// Since version 2010
public boolean GetEmbedCSSInHTML 1285();
public void SetEmbedCSSInHTML 1285( boolean i_bVal );
public boolean GetCreateDiagramsFolder 1284();
public void SetCreateDiagramsFolder 1284( boolean i_bVal );

// Since version 2010r3
public boolean GetGenerateRelativeLinks 1285();
public void SetGenerateRelativeLinks 1285( boolean i_bVal );

// Since version 2011r2
public boolean GetUseFixedDesign 1296();
public void SetUseFixedDesign 1296( boolean i_bVal );
public String GetSPSFile 1295();
public void SetSPSFile 1295( String i_strVal );
}
```

25.3.5.23 SpyTextImportExportSettings (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

(C) 2015-2021 Altova GmbH

For information about how to access the Application API from Java code, see the section: Programming Languages | Java[1118].

```
public class SpyTextImportExportSettings
{
   public void ReleaseInstance[1373]();
   public String GetDestinationFolder[1302]();
   public void SetDestinationFolder[1302]( String sVal );
   public long[1409] GetEnclosingCharacter[1302]();
   public void SetEnclosingCharacter[1302]( long[1409] nEnclosing );
   public String GetEncoding[1302]();
   public void SetEncoding[1302]( String sVal );
   public long[1401] GetEncodingByteOrder[1302]();
   public void SetEncodingByteOrder[1302]( long[1401] nByteOrder );
   public long[1408] GetFieldDelimiter[1303]();
   public void SetFieldDelimiter[1303]( long[1408] nDelimiter );
   public String GetFileExtension[1303] ();
   public void SetFileExtension[1303]( String sVal );
   public boolean GetHeaderRow[1303]();
   public void SetHeaderRow[1303]( boolean bVal );
   public String GetImportFile[1303]();
   public void SetImportFile[1303]( String sVal );
}
```

25.3.5.24 SpyTextView (obsolete)

```
public class SpyTextView
{
  public void ReleaseInstance[1373]();
  public SpyApplication[1375] GetApplication[1305]();
  public SpyDoc[1378] GetParent[1307]();
  public long LineFromPosition[1307]( long nCharPos );
  public long PositionFromLine[1308]( long nLine );
  public long LineLength[1307]( long nLine );
  public String GetSelText[1309]();
  public void SetSelText[1309]( String sText );
  public String GetRangeText[1306]( long nPosFrom, long nPosTill );
  public void ReplaceText[1308]( long nPosFrom, long nPosTill, String sText );
  public void MoveCaret[1307]( long nDiff );
  public void GoToLineChar[1306]( long nLine, long nChar );
  public void SelectText[1309]( long nPosFrom, long nPosTill );
  public long GetSelectionStart[1308]();
  public void SetSelectionStart[1308]( long nNewVal );
  public long GetSelectionEnd[1308]();
  public void SetSelectionEnd[1308]( long nNewVal );
  public String GetText[1309]();
  public void SetText[1309]( String sText );
  public long LineCount[1306]();
  public long Length[1306]();
}
```

25.3.5.25 SpyXMLData (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java [1118].

```
public class SpyXMLData
{
   public void ReleaseInstance[1373]();
   public void AppendChild[1311]( SpyXMLData[1392] oNewData );
   public void EraseAllChildren[1312]();
   public void EraseCurrentChild[1312]();
   public SpyXMLData[1392] GetCurrentChild[1315]();
   public SpyXMLData[1392] GetFirstChild[1315]( long[1411] nKind );
   public SpyXMLData[1392] GetNextChild[1316]();
   public boolean GetHasChildren[1317]();
   public void InsertChild[1318]( SpyXMLData[1392] oNewData );
   public boolean IsSameNode[1319]( SpyXMLData[1392] oToComp);
   public long[1411] GetKind[1319]();
   public boolean GetMayHaveChildren[1319]();
   public String GetName[1320]();
   public void SetName[1320]( String sValue );
   public SpyXMLData[1392] GetParent[1320]();
   public String GetTextValue[1320]();
   public void SetTextValue[1320]( String sValue );
}
```

25.3.5.26 Authentic (obsolete)

25.3.5.26.1 SpyAuthenticRange (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java [1118].

```
// Since version 2004R3
public class SpyAuthenticRange
{
    public void ReleaseInstance[1373]();
    public SpyApplication[1375] GetApplication[1159]();
    public SpyAuthenticView[1394] GetParent[1179]();
```

```
    public SpyAuthenticRange[1392] GotoNext[1169]( long[1398] eKind );
    public SpyAuthenticRange[1392] GotoPrevious[1170]( long[1398] eKind );
    public void Select[1181]();
    public long GetFirstTextPosition[1164]();
    public void SetFirstTextPosition[1164]( long nTextPosition );
    public long GetLastTextPosition[1175]();
    public void SetLastTextPosition[1175]( long nTextPosition );
    public String GetText[1184]();
    public void SetText[1184]( String strText );
    public boolean PerformAction[1180]( long[1397] eAction, String strElementName );
    public boolean CanPerformAction[1159]( long[1397] eAction, String strElementName );
    public String[] CanPerformActionWith[1160]( long[1397] eAction );
    public SpyAuthenticRange[1392] GoTo[1169]( long[1398] eKind, long nCount, long[1398] nFrom );
    public SpyAuthenticRange[1392] SelectNext[1181]( long[1398] eKind );
    public SpyAuthenticRange[1392] SelectPrevious[1182]( long[1398] eKind );
    public SpyAuthenticRange[1392] MoveBegin[1178]( long[1398] eKind, long nCount );
    public SpyAuthenticRange[1392] MoveEnd[1178]( long[1398] eKind, long nCount );
    public SpyAuthenticRange[1392] ExpandTo[1163]( long[1398] eKind );
    public SpyAuthenticRange[1392] CollapsToBegin[1160]();
    public SpyAuthenticRange[1392] CollapsToEnd[1161]();
    public SpyAuthenticRange[1392] GotoNextCursorPosition[1170]();
    public SpyAuthenticRange[1392] GotoPreviousCursorPosition[1171]();
    public boolean IsEmpty[1173]();
    public boolean IsEqual[1174]( SpyAuthenticRange[1392] ipCmp );
    public SpyAuthenticRange[1392] Clone[1160]();
    public SpyAuthenticRange[1392] SetFromRange[1184]( SpyAuthenticRange[1392] ipSrc );
    public boolean Delete[1161]();
    public boolean Cut[1161]();
    public boolean Copy[1161]();
    public boolean Paste[1179]();
    public SpyXMLData[1392] GetFirstXMLData[1164]();
    public void SetFirstXMLData[1164]( SpyXMLData[1392] objXMLDataPtr );
    public long GetFirstXMLDataOffset[1165]();
    public void SetFirstXMLDataOffset[1165]( long nOffset );
    public SpyXMLData[1392] GetLastXMLData[1176]();
    public void SetLastXMLData[1176]( SpyXMLData[1410] objXMLDataPtr );
    public long GetLastXMLDataOffset[1177]();
    public void SetLastXMLDataOffset[1177]( long nOffset );
    public String[] GetElementHierarchy[1167]();
    public String[] GetElementAttributeNames[1166]( String strElementName );
    public boolean HasElementAttribute[1171]( String strElementName, String strAttributeName
);
    public String GetElementAttributeValue[1167]( String strElementName, String
strAttributeName );
    public void SetElementAttributeValue[1183]( String strElementName, String
strAttributeName, String strNewValue );
    public String[] GetEntityNames[1168]();
    public void InsertEntity[1171]( String strEntityName );
    public boolean IsInDynamicTable[1174]();
    public boolean AppendRow[1158]();
    public boolean InsertRow[1172]();
    public boolean DuplicateRow[1162]();
    public boolean DeleteRow[1162]();
```

```
    public boolean MoveRowUp[1179] ();
    public boolean MoveRowDown[1179] ();

    // Since version 2004R4
    public boolean IsCopyEnabled[1173]();
    public boolean IsCutEnabled[1173]();
    public boolean IsPasteEnabled[1175]();
    public boolean IsDeleteEnabled[1173]();
    public boolean IsTextStateApplied[1175]( String i_strElementName );
    public boolean IsFirstRow[1174]();
    public boolean IsLastRow[1174]();
}
```

25.3.5.26.2 SpyAuthenticView (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java[1118].

```
// Since version 2004R3
public class SpyAuthenticView
{
    public void ReleaseInstance[1373] ();
    public SpyApplication[1375] GetApplication[1194] ();
    public SpyDoc[1378] GetParent[1200] ();
    public SpyAuthenticRange[1392] GetSelection[1201] ();
    public void SetSelection[1201] ( SpyAuthenticRange[1392] obj );
    public SpyAuthenticRange[1392] GetDocumentBegin[1196] ();
    public SpyAuthenticRange[1392] GetDocumentEnd[1196] ();
    public SpyAuthenticRange[1392] GetWholeDocument[1203] ();
    public long[1399] GetMarkupVisibility[1200] ();
    public void SetMarkupVisibility[1200] ( long[1399] eSpyAuthenticMarkupVisibility );
    public SpyAuthenticRange[1392] GoTo[1198] ( long[1398] eKind, long nCount, long[1398] nFrom );
    public void Print[1200] ( boolean bWithPreview, boolean bPromptUser );
    public boolean Undo[1202] ();
    public boolean Redo[1201] ();
    public void UpdateXMLInstanceEntities[1202] ();

    // Since version 2004R4
    public String GetAsXMLString[1195]();
    public void SetAsXMLString[1195]( String i_strXML );
    public SpyXMLData[1392] GetXMLDataRoot[1203]();
    public boolean IsUndoEnabled[1199]();
    public boolean IsRedoEnabled[1178]();
}
```

25.3.5.26.3 SpyDocEditSelection (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java [1118].

```
public class SpyDocEditSelection
{
   public void ReleaseInstance [1373]();
   public SpyXMLData [1392] GetEnd [1334]();
   public long GetEndTextPosition [1334]();
   public SpyXMLData [1392] GetStart [1335]();
   public long GetStartTextPosition [1335]();
}
```

25.3.5.26.4 SpyDocEditView (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java [1118].

```
public class SpyDocEditView
{
   public void ReleaseInstance [1373]();
   public void ApplyTextState [1337]( String sElementName );
   public SpyDocEditSelection [1395] GetCurrentSelection [1338]();
   public void EditClear [1338]();
   public void EditCopy [1339]();
   public void EditCut [1339]();
   public void EditPaste [1340]();
   public void EditRedo [1340]();
   public void EditSelectAll [1340]();
   public void EditUndo [1341]();
   public SpyXMLData [1392] GetNextVisible [1344]( SpyXMLData [1392] oElement );
   public SpyXMLData [1392] GetPreviousVisible [1344]( SpyXMLData [1392] oElement );
   public boolean GetIsEditClearEnabled [1345]();
   public boolean GetIsEditCopyEnabled [1345]();
   public boolean GetIsEditCutEnabled [1346]();
   public boolean GetIsEditPasteEnabled [1346]();
   public boolean GetIsEditRedoEnabled [1347]();
   public boolean GetIsEditUndoEnabled [1347]();
   public boolean GetIsRowAppendEnabled [1348]();
   public boolean GetIsRowDeleteEnabled [1348]();
   public boolean GetIsRowDuplicateEnabled [1349]();
```

```
  public boolean GetIsRowInsertEnabled[1349]();
  public boolean GetIsRowMoveDownEnabled[1350]();
  public boolean GetIsRowMoveUpEnabled[1350]();
  public boolean IsTextStateApplied[1351]( String sElementName );
  public boolean IsTextStateEnabled[1351]( String sElementName );
  public void LoadXML[1352]( String sXML );
  public void MarkUpView[1352]( long nKind );
  public void RowAppend[1353]();
  public void RowDelete[1353]();
  public void RowDuplicate[1354]();
  public void RowInsert[1354]();
  public void RowMoveDown[1355]();
  public void RowMoveUp[1355]();
  public String SaveXML[1356]();
  public void SelectionMoveTabOrder[1356]( boolean bForward, boolean bTag );
  public boolean SelectionSet[1357]( SpyXMLData[1392] oStart, long nStartPos, SpyXMLData[1392] oEndElement, long
  nEndPos );
  public SpyXMLData[1392] GetXMLRoot[1358]();
  public String[] GetAllowedElements[1342]( long nAction, SpyXMLData[1392] oStartPtr, SpyXMLData[1392] oEndPtr );
}
```

25.3.5.27 Predefined constants (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java[1118].

This section lists all classes that define the predefined constants used by the Java interface.

25.3.5.27.1 SPYApplicationStatus (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java[1118].

```
public class SPYApplicationStatus
  {
      public final static long spyApplicationStatus_Running                  = 0;
      public final static long spyApplicationStatus_AfterLicenseCheck        = 1;
      public final static long spyApplicationStatus_BeforeLicenseCheck       = 2;
      public final static long                                               = 3;
      spyApplicationStatus_ConcurrentLicenseCheckFailed
      public final static long spyApplicationStatus_ProcessingCommandLine    = 4;
  }
```

25.3.5.27.2 SPYAttributeTypeDefinition (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java 1118.

```
public class SPYAttributeTypeDefinition
{
   public final static long              = 0;
   spyMergedGlobal
   public final static long              = 1;
   spyDistinctGlobal
   public final static long spyLocal     = 2;
}
```

25.3.5.27.3 SPYAuthenticActions (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java 1118.

```
public class SPYAuthenticActions
{
    public final static long spyAuthenticInsertAt   = 0;
    public final static long spyAuthenticApply      = 1;
    public final static long spyAuthenticClearSurr  = 2;
    public final static long spyAuthenticAppend     = 3;
    public final static long                        = 4;
    spyAuthenticInsertBefore
    public final static long spyAuthenticRemove     = 5;
}
```

25.3.5.27.4 SPYAuthenticDocumentPosition (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java[1118].

```
public class SPYAuthenticDocumentPosition
{
    public final static long                        = 0;
    spyAuthenticDocumentBegin
    public final static long                        = 1;
    spyAuthenticDocumentEnd
    public final static long spyAuthenticRangeBegin = 2;
    public final static long spyAuthenticRangeEnd   = 3;
}
```

25.3.5.27.5 SPYAuthenticElementKind (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java[1118].

```
public class SPYAuthenticElementKind
{
   public final static long              = 0;
   spyAuthenticChar
   public final static long              = 1;
   spyAuthenticWord
   public final static long              = 3;
   spyAuthenticLine
   public final static long              = 4;
   spyAuthenticParagraph
   public final static long              = 6;
   spyAuthenticTag
   public final static long              = 8;
   spyAuthenticDocument
   public final static long              = 9;
   spyAuthenticTable
```


(C) 2015-2021 Altova GmbH

```
    public final static long                = 10;
    spyAuthenticTableRow
    public final static long                = 11;
    spyAuthenticTableColumn
}
```

25.3.5.27.6 SPYAuthenticMarkupVisibility (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java 1118.

```
public class SPYAuthenticMarkupVisibility
{
    public final static long                        = 0;
    spyAuthenticMarkupHidden
    public final static long spyAuthenticMarkupSmall= 1;
    public final static long spyAuthenticMarkupLarge= 2;
    public final static long spyAuthenticMarkupMixed= 3;
}
```

25.3.5.27.7 SPYDatabaseKind (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java 1118.

```
public class SPYLoading
 {
    public final static long spyDB_Access   = 0;
    public final static long                = 1;
    spyDB_SQLServer
    public final static long spyDB_Oracle   = 2;
    public final static long spyDB_Sybase   = 3;
    public final static long spyDB_MySQL    = 4;
    public final static long spyDB_DB2      = 5;
    public final static long spyDB_Other    = 6;
    public final static long                = 7;
    spyDB_Unspecified
```

```
}
```

25.3.5.27.8 SPYDialogAction (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java[1118].

```
public class SPYDialogAction
 {
     public final static long spyDialogOK    = 0;
     public final static long                = 1;
     spyDialogCancel
     public final static long                = 2;
     spyDialogUserInput
 }
```

25.3.5.27.9 SPYDOMType (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java[1118].

```
public class SPYDOMType
 {
     public final static long                = 0;
     spyDOMType_msxml4
     public final static long                = 1;
     spyDOMType_xerces
 }
```

25.3.5.27.10 SPYDTDSchemaFormat (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

(C) 2015-2021 Altova GmbH

For information about how to access the Application API from Java code, see the section: Programming Languages | Java 1118.

```
public class SPYDTDSchemaFormat
{
    public final static long spyDTD     = 0;
    public final static long spyDCD     = 1;
    public final static long spyXMLData = 2;
    public final static long spyBizTalk = 3;
    public final static long spyW3C     = 4;
}
```

25.3.5.27.11 SPYEncodingByteOrder (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java 1118.

```
public class SPYEncodingByteOrder
{
    public final static long spyNONE          = 0;
    public final static long spyLITTLE_ENDIAN = 1;
    public final static long spyBIG_ENDIAN    = 2;
}
```

25.3.5.27.12 SPYExportNamespace (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java 1118.

```
public class SPYExportNamespace
{
    public final static long spyNoNamespace                = 0;
    public final static long spyReplaceColonWithUnderscore = 1;
}
```

25.3.5.27.13 SPYFindInFilesSearchLocation (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java 1118.

```
public class SPYFindInFilesSearchLocation
{
   public final static long                    = 0;
   spyFindInFiles_Documents
   public final static long                    = 1;
   spyFindInFiles_Project
   public final static long                    = 2;
   spyFindInFiles_Folder
}
```

25.3.5.27.14 SPYFrequentElements (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java 1118.

```
public class SPYFrequentElements
{
   public final static long                = 0;
   spyGlobalElements
   public final static long                = 1;
   spyGlobalComplexType
}
```

25.3.5.27.15 SPYImageKind (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

(C) 2015-2021 Altova GmbH

For information about how to access the Application API from Java code, see the section: Programming Languages | Java 1118.

```
public class SPYImageKind
 {
     public final static long                = 0;
     spyImageType_PNG
     public final static long                = 1;
     spyImageType_EMF
 }
```

25.3.5.27.16 SPYImportColumnsType (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java 1118.

25.3.5.27.17 SPYLibType (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java 1118.

```
public class SPYLibType
 {
     public final static long                = 0;
     spyLibType_static
     public final static long spyLibType_dll= 1;
 }
```

25.3.5.27.18 SPYLoading (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java[1118].

```
public class SPYLoading
{
    public final static long                = 0;
    spyUseCacheProxy
    public final static long spyReload      = 1;
}
```

25.3.5.27.19 SPYNumberDateTimeFormat (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java[1118].

```
public class  SPYNumberDateTimeFormat
{
    public final static long spySystemLocale = 0;
    public final static long                 = 1;
    spySchemaCompatible
}
```

25.3.5.27.20 SPYProgrammingLanguage (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java[1118].

```
public class SPYLoading
 {
```


(C) 2015-2021 Altova GmbH

```
    public final static long spyUndefinedLanguage = -1;
    public final static long spyJava              = 0;
    public final static long spyCpp               = 1;
    public final static long spyCSharp            = 2;
}
```

25.3.5.27.21 SPYProjectItemTypes (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java 1118.

```
public class SPYProjectItemTypes
{
    public final static long            = 0;
    spyUnknownItem
    public final static long spyFileItem= 1;
    public final static long            = 2;
    spyFolderItem
    public final static long spyURLItem = 3;
}
```

25.3.5.27.22 SPYProjectType (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java 1118.

```
public class SPYProjectType
{
    public final static long                  = 0;
    spyVisualStudioProject
    public final static long                  = 1;
    spyVisualStudio2003Project
    public final static long spyBorlandProject = 2;
    public final static long spyMonoMakefile   = 3;
}
```

25.3.5.27.23 SPYSampleXMLGenerationOptimization (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java 1118.

```
public class SPYSampleXMLGenerationOptimization
{
   public final static long spySampleXMLGen_Optimized  = 0;
   public final static long                            = 1;
   spySampleXMLGen_NonMandatoryElements
   public final static long spySampleXMLGen_Everything = 2;
}
```

25.3.5.27.24 SPYSampleXMLGenerationSchemaOrDTDAssignment (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java 1118.

```
public class SPYSampleXMLGenerationOptimization
{
   public final static long                            = 0;
   spySampleXMLGen_AssignRelatively
   public final static long                            = 1;
   spySampleXMLGen_AssignAbsolutely
   public final static long spySampleXMLGen_DoNotAssign= 2;
}
```

25.3.5.27.25 SPYSchemaDefKind (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

(C) 2015-2021 Altova GmbH

For information about how to access the Application API from Java code, see the section: Programming Languages | Java[1118].

```
public class SPYSchemaDefKind
{
    public final static long spyKindElement        = 0;
    public final static long spyKindComplexType    = 1;
    public final static long spyKindSimpleType     = 2;
    public final static long spyKindGroup          = 3;
    public final static long spyKindModel          = 4;
    public final static long spyKindAny            = 5;
    public final static long spyKindAttr           = 6;
    public final static long spyKindAttrGroup      = 7;
    public final static long spyKindAttrAny        = 8;
    public final static long spyKindIdentityUnique= 9;
    public final static long spyKindIdentityKey    = 10;
    public final static long spyKindIdentityKeyRef= 11;
    public final static long                       = 12;
    spyKindIdentitySelector
    public final static long spyKindIdentityField = 13;
    public final static long spyKindNotation       = 14;
    public final static long spyKindInclude        = 15;
    public final static long spyKindImport         = 16;
    public final static long spyKindRedefine       = 17;
    public final static long spyKindFacet          = 18;
    public final static long spyKindSchema         = 19;
    public final static long spyKindCount          = 20;
}
```

25.3.5.27.26 SPYSchemaDocumentationFormat (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java[1118].

```
public class SPYSchemaDocumentationFormat
 {
    public final static long                = 0;
    spySchemaDoc_HTML
    public final static long                = 1;
    spySchemaDoc_MSWord
    public final static long spySchemaDoc_RTF        = 2;
    public final static long spySchemaDoc_PDF        = 3;
 }
```

25.3.5.27.27 SPYSchemaExtensionType (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java[1118].

```
public class SPYSchemaExtensionType
 {
    public final static long spySchemaExtension_None    = 0;
    public final static long spySchemaExtension_SQL_XML = 1;
    public final static long                            = 2;
    spySchemaExtension_MS_SQL_Server
    public final static long spySchemaExtension_Oracle  = 3;
 }
```

25.3.5.27.28 SPYSchemaFormat (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java[1118].

```
public class SPYSchemaFormat
 {
    public final static long                        = 0;
    spySchemaFormat_Hierarchical
    public final static long spySchemaFormat_Flat   = 1;
 }
```

25.3.5.27.29 SPYTextDelimiters (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java[1118].

(C) 2015-2021 Altova GmbH

```
public class SPYTextDelimiters
{
    public final static long spyTabulator = 0;
    public final static long spySemicolon = 1;
    public final static long spyComma     = 2;
    public final static long spySpace     = 3;
}
```

25.3.5.27.30 SPYTextEnclosing (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java 1118.

```
public class SPYTextEnclosing
{
    public final static long          = 0;
    spyNoEnclosing
    public final static long          = 1;
    spySingleQuote
    public final static long          = 2;
    spyDoubleQuote
}
```

25.3.5.27.31 SPYTypeDetection (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java 1118.

```
public class SPYTypeDetection
{
    public final static long              = 0;
    spyBestPossible
    public final static long              = 1;
    spyNumbersOnly
    public final static long              = 2;
    spyNoDetection
}
```

25.3.5.27.32 SPYURLTypes (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java 1118.

```
public class SPYURLTypes
{
    public final static long              = (-1);
    spyURLTypeAuto
    public final static long              = 0;
    spyURLTypeXML
    public final static long              = 1;
    spyURLTypeDTD
}
```

25.3.5.27.33 SpyViewModes (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java 1118.

```
public class SPYViewModes
{
    public final static long spyViewGrid     = 0;
    public final static long spyViewText     = 1;
    public final static long spyViewBrowser  = 2;
    public final static long spyViewSchema   = 3;
    public final static long spyViewContent  = 4;
    public final static long                 = 4;
    spyViewAuthentic
    public final static long spyViewWSDL     = 5;
    public final static long spyViewZIP      = 6;
    public final static long                 = 7;
    spyViewEditionInfo
}
```


(C) 2015-2021 Altova GmbH

25.3.5.27.34 SPYWhitespaceComparison (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java 1118.

```
public class SPYWhitespaceComparison
 {
    public final static long spyCompareAsIs  = 0;
    public final static long                 = 1;
    spyCompareNormalized
    public final static long spyStripAll     = 2;
 }
```

25.3.5.27.35 SPYXMLDataKind (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java 1118.

```
public class SPYXMLDataKind
{
   public final static long                  = 0;
   spyXMLDataXMLDocStruct
   public final static long                  = 1;
   spyXMLDataXMLEntityDocStruct
   public final static long                  = 2;
   spyXMLDataDTDDocStruct
   public final static long spyXMLDataXML    = 3;
   public final static long                  = 4;
   spyXMLDataElement
   public final static long spyXMLDataAttr   = 5;
   public final static long spyXMLDataText   = 6;
   public final static long spyXMLDataCData  = 7;
   public final static long                  = 8;
   spyXMLDataComment
   public final static long spyXMLDataPI     = 9;
   public final static long                  = 10;
   spyXMLDataDefDoctype
```

```
    public final static long                    = 11;
    spyXMLDataDefExternalID
    public final static long                    = 12;
    spyXMLDataDefElement
    public final static long                    = 13;
    spyXMLDataDefAttlist
    public final static long                    = 14;
    spyXMLDataDefEntity
    public final static long                    = 15;
    spyXMLDataDefNotation
    public final static long                    = 16;
    spyXMLDataKindsCount
}
```


(C) 2015-2021 Altova GmbH

25.4 ActiveX Integration

このセクションで説明されているXMLSpy ユーザーインターフェイスとActiveX コントロール機能を使用するアプリケーションに統合することができます。 ActiveX 技術によりC++、C#、VB.NET、HTML などの統合のために使用される広範囲の言語が有効化されます(HTML 内に統合されているActiveX コンポーネントは、Microsoft Internet Explorer バージョンとActiveX をサポートすると共に作動する必要があります)。全てのコンポーネントはフルOLE コントロールです。Java への統合はラッパークラスにより与えられます。

> ActiveX コントロールを、カスタムコードに統合するには、XMLSpy Integration Package をインストールしてください(次を参照してください: http://www.altova.com/ide_integration.html)。すべてのXMLSpy を最初に、そして、XMLSpy Integration Package そして他の必要条件を言語とプラットフォームに応じてインストールしてください(次を参照してください: 必要条件[1413])。

統合の2つの異なるレベル間を柔軟に選択することができます: アプリケーションレベルとドキュメントレベル.

アプリケーションレベルの統合は、(メニュー、ツールバー、ペインを含む) XMLSpy のフルインターフェイスを ActiveX コントロールをカスタムアプリケーションとして埋め込むことを意味します。例えば、多くのシナリオでは、カスタムアプリケーションを、XMLSpy グラフィカルユーザーインターフェイスを埋め込む1つのフォームから構成することができます。

ドキュメントレベルの統合よりもこのアプローチより簡単に実装することができますが、カスタム必要条件に従いを柔軟的にXMLSpy グラフィカルユーザーインターフェイスを構成するためには適切ではない可能性があります。ドキュメントレベルの統合は、アプリケーションにXMLSpy を1つ埋め込むことを意味します。メインXMLSpy コントロールのみではなく、メインドキュメント エディターウィンドウの実装も含まれます。また、オプションで、追加ウィンドウの実装も含まれます。このアプローチはGUI の構成に柔軟性を与えますが、選択された言語によりActiveX コントロールとの高度なインタラクションが必要とされます。

セクションアプリケーションレベルの統合[1416]とドキュメントレベルの統合[1418] は、それぞれのレベルの主要なステップに関して説明しています。ActiveX 統合のサンプル[1421] セクションは、C#、HTML、および、Java のサンプルを提供します。これらのサンプルを確認することにより、正しい選択が直ぐに行うことができます。オブジェクトレファレンス[1459] セクションは、プロパティとメソッドに加え統合のために使用できるすべてのCOM オブジェクトに付いて説明しています。

XMLSpy をVisual Studio プラグインとして使用するための情報に関しては、次を参照してください: Visual Studio 内のXMLSpy[729]。

25.4.1 必要条件

XMLSpy ActiveX コントロールをカスタムアプリケーションに統合するには、以下が使用中のコンピューターにインストールされている必要があります:

- XMLSpy
- XMLSpy Integration Package は、次からダウンロードできます: http://www.altova.com/ide_integration.html

64 ビット ActiveX コントロールを統合するには、XMLSpy の64 ビット バージョンとXMLSpy Integration Package をインストールする必要があります。Visual Studio を使用して、Microsoft .NET プラットフォームの下で開発されたアプリケーションに関しては、XMLSpy の32 ビット と64 ビット バージョンとXMLSpy Integration Package について下で説明されているとおりインストールします。

メモ Windows XP とWindows Server 2003 で統合パッケージが正確に作動するためには、再頒布可能パッケージを手動でインストールしてください: https://www.microsoft.com/en-US/download/details.aspx?id=48145.

Microsoft .NET (C#, VB.NET) with Visual Studio

XMLSpy ActiveX コントロールを、Microsoft .NET の下で開発された 32 ビット アプリケーションに統合するには、以下を使用中のコンピューターにインストールしてください:

- Microsoft .NET Framework 4.0 または以降
- Visual Studio 2010/2012/2013/2015/2017/2019
- XMLSpy 32 ビットとXMLSpy Integration Package 32 ビット
- ActiveX コントロールがVisual Studio ツールボックスに追加されている必要があります（次を参照してください: ActiveX コントロールをツールボックスに追加する[1414]).

64 ビット ActiveX コントロールを統合する場合、次の必要条件が上記に加え満たされる必要があります:

- XMLSpy 32 ビットとXMLSpy Integration Package 32 ビット をインストールしてください（Visual Studio は32 ビットで作動するため、32 ビット ActiveX コントロールを Visual Studio デザイナーに与える必要があります。）
- XMLSpy 64 ビットとXMLSpy Integration Package 64 ビット をインストールしてください（実際の64 ビット ActiveX コントロールをランタイムにカスタムアプリケーションに与えます。）
- Visual Studio 内で、64 ビットビルド更新を作成し、この更新を使用してアプリケーションをビルドします。サンプルに関しては、次を参照してください: サンプルC# ソリューションの実行[1421]。

Java

Eclipse 開発環境を使用して、XMLSpy ActiveX コントロールを Java アプリケーションに統合するには、次を使用中のコンピューターにインストールしてください:

- Java Runtime Environment（JRE）または Java Development Kit（JDK）7 または以降
- Eclipse
- XMLSpy and XMLSpy Integration Package

メモ XMLSpy ActiveX コントロールの64 ビット バージョンを作動するには、Eclipse の64 ビット バージョン、および、XMLSpy とXMLSpy Integration Package の64 ビット バージョン ofを使用してください。

クライアントコンピューター上でのXMLSpy 統合とデプロイ

.NET アプリケーションを作成し他のクライアントに配布する場合は、クライアントコンピューター上に次をインストールしてください:

- XMLSpy
- XMLSpy Integration Package
- カスタムの統合コード、または、アプリケーション。

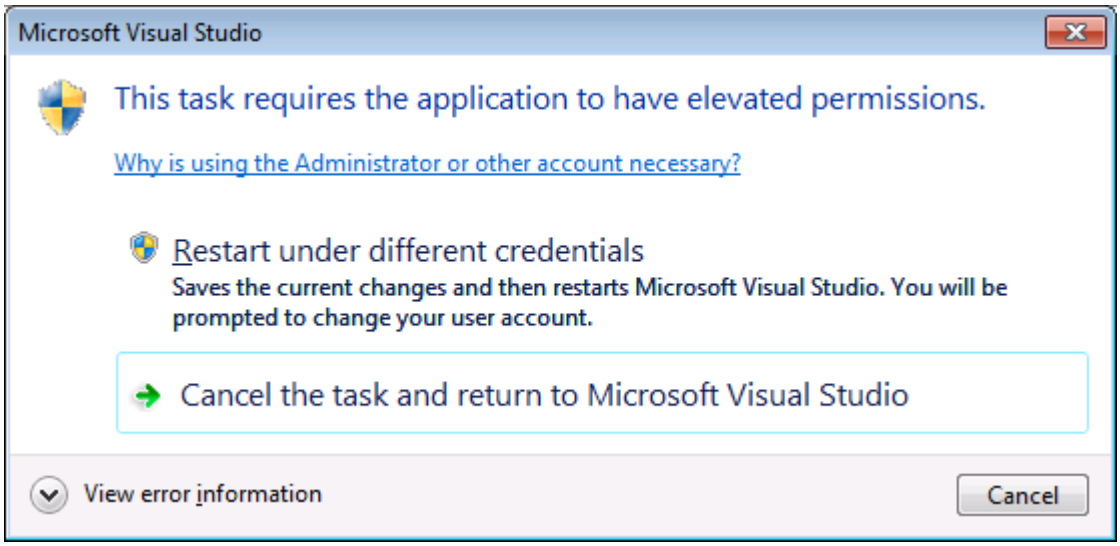
25.4.2 ActiveX コントロールをツールボックスに追加する

Visual Studio を使用して開発されたアプリケーション内で、XMLSpy ActiveX コントロールを使用するには、コントロールが最初に Visual Studio ツールボックスに追加される必要があります:

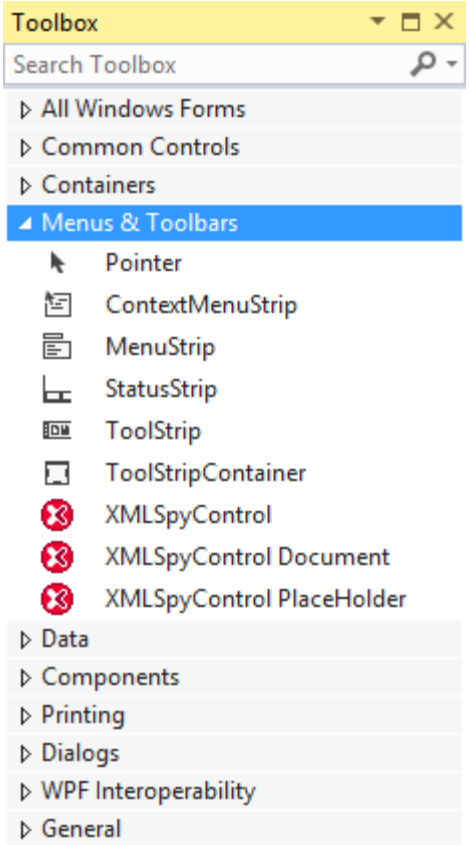
1. Visual Studio の「ツール」メニューから、「ツールボックスアイテムの選択」をクリックします。
2. 「COM コンポーネント」タブ、から、XMLSpyControl, XMLSpyControl ドキュメント、および、XMLSpyControl プレースホルダーの横のチェックボックスを選択します。

上記のコントロールが使用することができない場合、下のステップに従います:

1. 「COM コンポーネント」タブから「参照」をクリックします。**XMLSpyControl.ocx** をXMLSpy インストールからフォルダーを選択します。XMLSpy Integration Package をインストールすることを忘れないでください。それ以外の場合、このファイルを使用することはできません。次を参照してください: 必要条件[1413].
2. 更に高いパーミッションと共に再起動するようにプロンプトされると、「異なる資格情報下で再起動する」をクリックします。



上記のステップが完了すると、XMLSpy ActiveX コントロールは、Visual Studio ツールボックス内で使用できるようになります。



メモ　アプリケーションレベルの統合では、**XMLSpyControl** ActiveX コントロールのみが使用されます（次を参照してください: アプリケーションレベルの統合[1416]）。 XMLSpyControl ドキュメント とXMLSpyControl プレースホルダー コントロールは、ドキュメントレベルの統合のために使用されます（次を参照してください: ドキュメントレベルの統合[1418]）。

25.4.3 アプリケーションレベルの統合

アプリケーションレベルの統合により使用中のアプリケーションのウィンドウにXMLSpy のインターフェイスを埋め込むことができます。 この種類の統合により、すべてのメニュー、ツールバー、ステータスバー、ドキュメントウィンドウ、および、ヘルパーウィンドウを含むXMLSpy のユーザーインターフェイスを使用することができます。アプリケーションのユーザーインターフェイスのカスタム化はXMLSpy が提供する機能に限られています。これには、ヘルパーウィンドウの整理とサイズ調整、及びメニューとツールバーのカスタム化が含まれています。

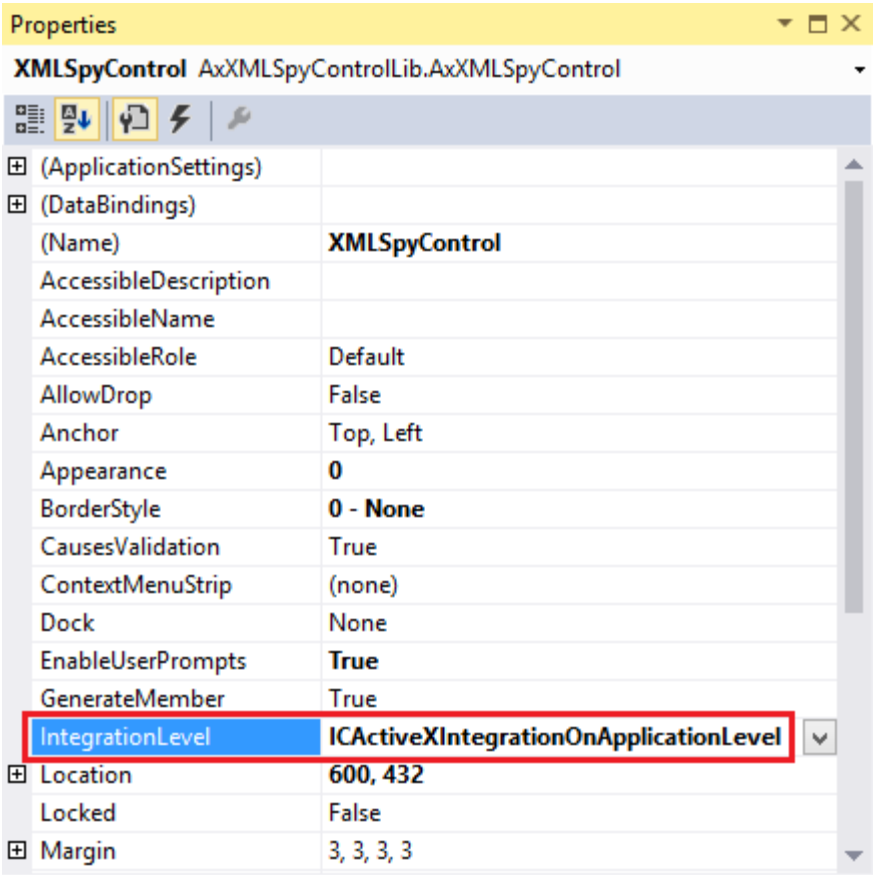
アプリケーションレベルで統合する場合、統合する必要のある唯一のActiveX コントロールは`XMLSpyControl`[1462] です。`XMLSpyControlDocument`[1470] または`XMLSpyControlPlaceHolder`[1476] ActiveX コントロールをインスタンス化、または、アクセス使用としないでください。

初期化、または、XMLSpy の振る舞いを自動化するには、`XMLSpyControl`[1462]. のために説明されているプロパティ、メソッド、とイベントを使用してください。XMLSpy 機能への複雑なアクセスのために`XMLSpyControl.application`[1463] を使用することを考慮してください。

XMLSpy アプリケーションをHTML ページ、に埋め込むサンプルに関しては、次を参照してください: アプリケーションレベルのHTML 統合[1424]。

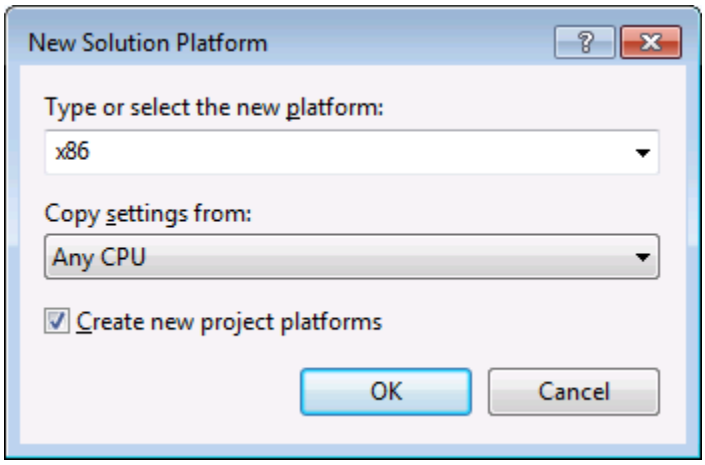
Visual Studio を使用する C# または VB.NET での XMLSpy ActiveX コントロールをアプリケーションレベルで統合する基本的なアプリケーションを作成するステップは、以下のとおりです:

1. すべての必要条件が満たされていることを確認してください（次を参照してください: 必要条件[1413]）.
2. 新規の Visual Studio Windows Forms プロジェクトを新規の空のフォーム共に作成してください。
3. まだ行っていない場合、ActiveX コントロールをツールボックスに追加します（次を参照してください: ActiveX コントロールをツールボックスに追加する[1414]）。
4. ツールボックスから、**XMLSpyControl** をドラッグして、新規のフォームにドロップしてください。
5. フォームで **XMLSpyControl** を選択し、プロパティウインドウ内で **IntegrationLevel** プロパティを **ICActiveXIntegrationOnApplicationLevel** に設定します。



6. ビルド（x86、x64）するプラットフォームに一致するビルドプラットフォームの更新を作成します。ビルドの更新の作成の方法は以下のとおりです:

   a. Visual Studio 内のソリューションを右クリックし、「**更新 マネージャー**」を選択します。
   b. 「**アクティブなソリューションプラットフォーム**」から、「**新規作成...**」を選択し、x86 または x64 更新を選択します（このサンプルでは、**x86**）。



Visual Studio 内でソリューションをビルドして実行する準備が整うと、ターゲットプラットフォーム(x86、x64)に一致する更新をしようしてビルドすることを確認してください。

25.4.4 ドキュメントレベルの統合

アプリケーションレベルの統合に比べ、ドキュメントレベルの統合複雑ですが、ActiveX コントロール。を使用することによりアプリケーションに XMLSpy 機能を柔軟的に埋め込むことができます。このアプローチにより、コードは XMLSpy ユーザーインターフェイスの次のパーツに部分的にアクセスすることができます:

- ドキュメントの編集ウィンドウ
- プロジェクトウィンドウ
- 入力ヘルパーウィンドウ
- バリデーター出力ウィンドウ
- XPath プロファイラーウィンドウ
- XPath ダイアログウィンドウ
- XSLT/XQuery デバッガーウィンドウ

アプリケーションレベルの統合[1416] で説明されているとおり、ActiveX アプリケーションレベルの統合のためには、**XMLSpyControl** のみが必要とされる唯一のコントロールです。しかしながら、ActiveX ドキュメントレベルの統合の場合は、機能 XMLSpy が次の ActiveX コントロールにより与えられます:

1. **XMLSpyControl**[1462]
2. XMLSpyControl ドキュメント[1470]
3. XMLSpyControl プレースホルダ[1476]

これらのコントロールは XMLSpy のアプリケーション インストールフォルダーで使用することのできる **XMLSpyControl.ocx** ファイルにより与えられます。Visual Studio を使用して ActiveX 統合を開発すると、これらのコントロールを Visual Studio ツールボックスに追加する必要があります(次を参照してください: ActiveX コントロールをツールボックスに追加する[1414])。

ドキュメントレベルでの ActiveX コントロールのアプリケーションへの統合の基本のステップは、以下のとおりです:

1. 最初に、アプリケーション内で **XMLSpyControl** をインスタンス化します。「**XMLSpyControl** ドキュメント」と「**XMLSpyControl** プレースホルダ」コントロールへのサポートが有効化されるために、このコントロールの統合は必須です。`IntegrationLevel`[1464] プロパティを「ICActiveXIntegrationOnDocumentLevel」(または「1」)に設定することは重要です。ユーザーに対してコントロールを非表示にするには、「Visible」プロパティを「False」に設定します。

メモ ドキュメントレベルで統合する場合、未知の結果が生成される可能性があるため、**XMLSpyControl** の「開く」メソッドを使用しないでください。**XMLSpyControl ドキュメント** と**XMLSpyControl プレースホルダ** の対応するメソッドを代わりに使用して開いてください。

2. **XMLSpyControl ドキュメント** のインスタンスを少なくとも1つアプリケーション内に作成します。このコントロールは、アプリケーションにXMLSpy のドキュメントの編集ウィンドウを与え、必要であれば、複数回インスタンス化することができます。

   既存のファイルを開くために、メソッド「開く」を使用します。ドキュメントに関連した機能にアクセスするには、メソッドとプロパティ「ドキュメント」を介してアクセスすることのできる「パス」と「保存」を使用します。

メモ コントロールは、読み取り専用モードをサポートしません。プロパティ「読み取り専用」の値は無視されます。

3. 任意で、アプリケーションに、アプリケーションで使用することのできる(ドキュメント ウィンドウ以外の)各追加ウィンドウのために **XMLSpyControl プレースホルダ** コントロールを追加することができます。

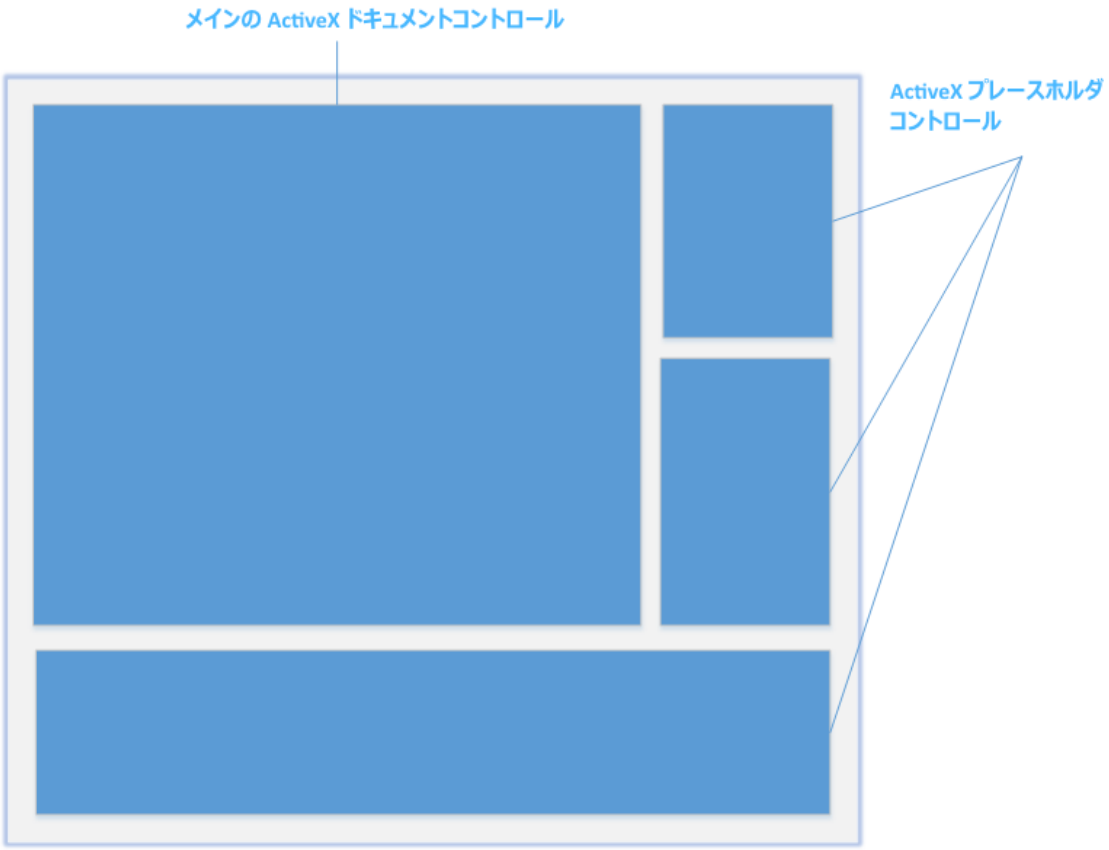
   XMLSpyControl プレースホルダ のインスタンスによりアプリケーションウィンドウに選択して XMLSpy の追加ウィンドウを埋め込むことができます。ウィンドウの種類(例えば、プロジェクトウィンドウ)はプロパティ「PlaceholderWindowID」により定義することができます。有効なウィンドウの識別子のためには、次を参照してください: XMLSpyControlPlaceholderWindow[1479].

メモ 各ウィンドウ識別子のために「**XMLSpyControl プレースホルダ**」のみを使用してください。

   XMLSpy プロジェクトウィンドウを選択するプレースホルダーコントロールのために、追加メソッドを使用することができます。「OpenProject」を使用して、XMLSpy プロジェクトをロードすることができます。XMLSpy 自動化インターフェイスからプロパティプロジェクトとメソッドを使用して、他のプロジェクトに関連したオペレーションを操作します。

例えば、Visual Studio を使用した C# または VB.NET 内では、XMLSpy ActiveX コントロールをドキュメントレベルで統合する基本的なアプリケーションの作成ステップは、以下にリストされるものに類似しています。必要に応じて適用は複雑になる可能性があります。しかしながら、ActiveX ドキュメントレベルの統合のための必要条件として、下の命令を理解することは重要です。

1. 新規の空のフォームと共に新規の Visual Studio Windows Forms プロジェクトを作成します。
2. まだ行っていない場合、ActiveX コントロールをツールボックスに追加します(次を参照してください: ActiveX コントロールをツールボックスに追加する[1414])。
3. ツールボックスから **XMLSpyControl**[1462] をドラッグして新規のフォームにドラッグします。
4. **XMLSpyControl** の「IntegrationLevel」プロパティを「ICActiveXIntegrationOnDocumentLevel」に、と「Visible」プロパティを「False」に設定します。コードから、または、「プロパティ」ウィンドウから行うことができます。
5. XMLSpyControl ドキュメント[1470] をツールボックスからフォームにドラッグします。このコントロールは、使用中のアプリケーションにXMLSpy のメインウィンドウを与え、ドキュメントのためにサイズを調整することができます。
6. 代わりに、フォームに1つまたは複数の XMLSpyControl プレースホルダ[1476] コントロールを追加することができます(アプリケーションが必要とする各追加ウィンドウの型など。例えば、「プロジェクト」ウィンドウなど)。このような追加プレースホルダーコントロールをメインドキュメントコントロールの下、または、左右に通常配置します。例:



7. 各「XMLSpyControl プレースホルダ」コントロールの「PlaceholderWindowID」プロパティを有効なウィンドウ識別子に設定してください。有効な値のリストに関しては、次を参照してください: XMLSpyControlPlaceholderWindow[1479]。
8. アプリケーションにコマンドを以下のように追加します(少なくとも、ドキュメントを開き、保存し、閉じる必要があります)。

XMLSpy コマンドをクエリする

ドキュメントレベルで統合する場合は、XMLSpy メニューまたはツールバーをアプリケーションで使用することはできません。代わりに、必要とするコマンドを取得し、状態を確認し、プログラム的に実行することができます:

- 使用することのできるすべてのコマンドを取得するには、**XMLSpyControl** の `CommandsList`[1464] プロパティを使用します。
- メニュー構造により整理されているコマンドを取得するには、メイン `Menu`[1465] プロパティを使用します。
- 表示される順番に整理されているコマンドを取得するには、`Toolbars`[1465] プロパティを使用します。
- XMLSpy にコマンドを送信するには、`Exec`[1466] メソッドを使用します。
- コマンドが現在有効化されているか、または、無効化されているかクエリするために、`QueryStatus`[1467] メソッドを使用します。

これにより XMLSpy コマンドをアプリケーションのメニューとツールバーに柔軟的に統合することができます。

XMLSpy のインストールは、XMLSpy 内で使用されているコマンドラベルを与えます。GIF 書式内のアイコンのために XMLSpy インストールのフォルダー「<ApplicationFolder>¥Examples¥ActiveX¥Images」を参照してください。fコマンド名に対応するファイル名は、コマンド参照[1440] セクション内にリストされています。

一般的な注意点

XMLSpy の振る舞いを自動化するには、**XMLSpyControl**[1462]、XMLSpyControl ドキュメント[1470] とXMLSpyControl プレースホルダ[1476] 内で説明されている プロパティ、メソッド、とイベントを使用してください。

XMLSpy 機能への高度なアクセスに関しては、次のプロパティを考慮してください:

- XMLSpyControl.Application[1463]
- XMLSpyControlDocument.Document[1471]
- XMLSpyControlPlaceHolder.Project[1477]

これらのプロパティはXMLSpy 自動化インターフェイス(XMLSpyAPI)へのアクセスを提供します

メモ ドキュメントを開くには、適切なドキュメントコントロールのXMLSpyControlDocument.Open[1473] または XMLSpyControlDocument.New[1473] を使用してください。 プロジェクトを開くには、XMLSpy プロジェクトウィンドウを埋め込むプレースホルダコントロール上のXMLSpyControlPlaceHolder.OpenProject[1478] を常に使用してください。

異なるプログラミング環境内で必要なコントロールをインスタンス化しアクセスする方法を説明するサンプルに関しては、次を参照してください: ActiveX 統合のサンプル[1421]。

25.4.5 ActiveX 統合のサンプル

このセクションは、異なるコンテナー環境とプログラム言語を使用するXMLSpy ドキュメントレベルの統合のサンプル、を含みます(HTML セクションには、追加でアプリケーションレベルの統合のサンプルが含まれます)。全てのサンプルのソースコードは、XMLSpy インストール内のフォルダー `<ApplicationFolder>\Examples\ActiveX` で使用することができます。

25.4.5.1 C#

C# and Visual Studio のための基本のActiveX 統合サンプルソリューションは、フォルダー `<ApplicationFolder>\Examples\ActiveX\C#` 内にあります。ソースコードをコンパイルする前に、サンプルを実行して、全ての必要条件が満たされていることを確認してください(次を参照してください: サンプルC# ソリューションの実行[1421])。

25.4.5.1.1 サンプル C# ソリューションの実行

フォルダー**<ApplicationFolder>\Examples\ActiveX\C#** 内で使用することのできるサンプルVisual Studio ソリューションは、XMLSpy ActiveX コントロールの使用方法について説明しています。このソリューションをビルドして実行する前に、次のステップに注意してください:

ステップ 1: 必要条件をチェックする

Visual Studio 2010 または以降は、サンプルソリューションを開く必要があります。必要条件の完全なリストは、次を参照してください: 必要条件[1413]。

ステップ 2: 書き込みのパーミッションを保有するディレクトリにサンプルをコピーする

Visual Studio を管理者として作動することを回避するには、デフォルトの場所から実行する代わりに、書き込みのパーミッションを持つディレクトリへのソースコードをコピーします。

ステップ 3: すべての必要とされるコントロールプロパティをチェックし設定する

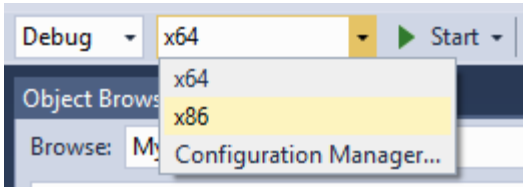
サンプルアプリケーションは、XMLSpyControlDocument[1470] の1つのインスタンスとXMLSpyControlPlaceHolder[1476] コントロールの1つのインスタンスを含んでいます。下のテーブルに表示されているとおりにこれらのコントロールの次のプロパティが設定されていることを確認してください:

コントロール名	プロパティ	プロパティ値
XMLSpyControl	IntegrationLevel	ICActiveXIntegrationOnDocumentLevel
XPathDialog	PlaceholderWindowID	16

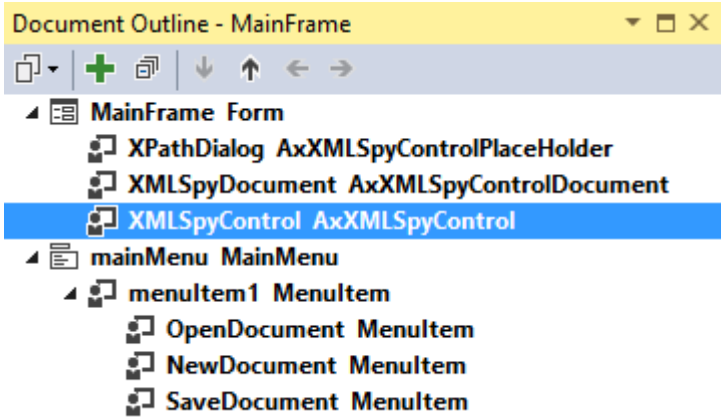
ActiveX コントロールのプロパティをビュー、または、設定する方法は、以下のとおりです:

1. デザイナーウィンドウ内で **MDIMain.cs** フォームを開きます。

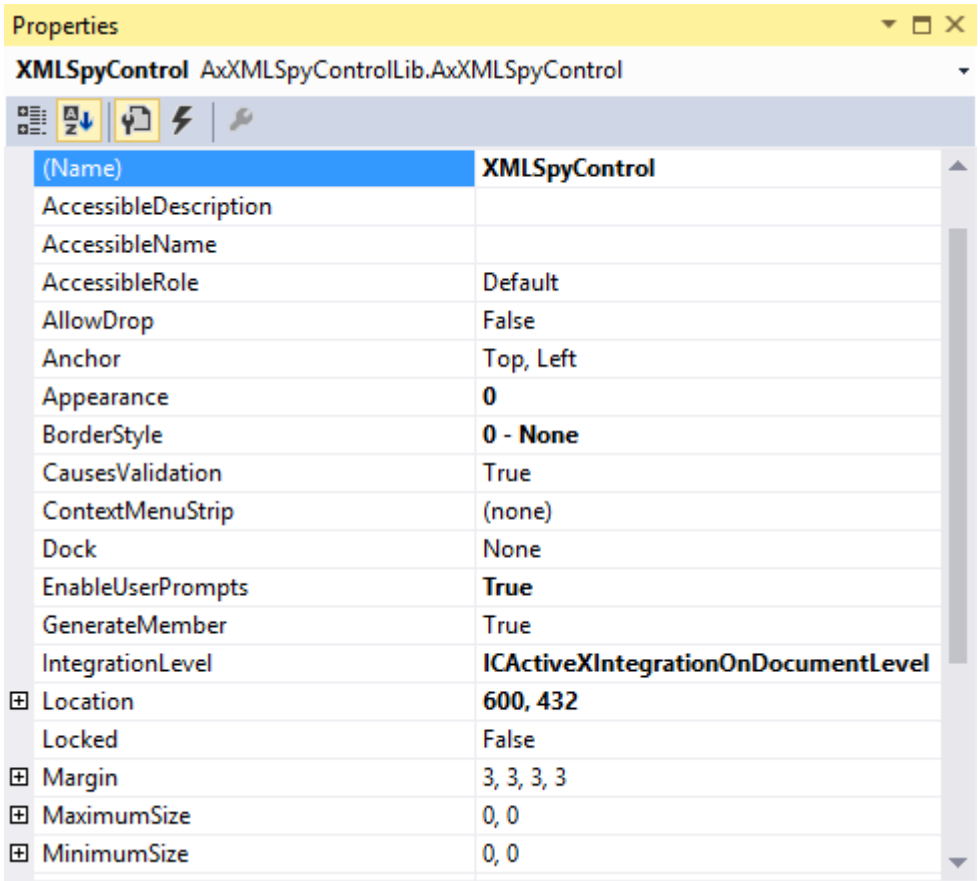
メモ 64 ビット Windows では、デザイナーウィンドウで開く前にVisual Studio ソリューションを「x86」にビルドを更新する必要がある可能性があります。64 ビットアプリケーションとしてサンプルをビルドする場合は、次を参照してください: 必要条件[1413].



2. Visual Studio のドキュメント概要 ウィンドウを開きます(「表示」メニューから、「他のウィンドウ | ドキュメントの概要」をクリックします)。



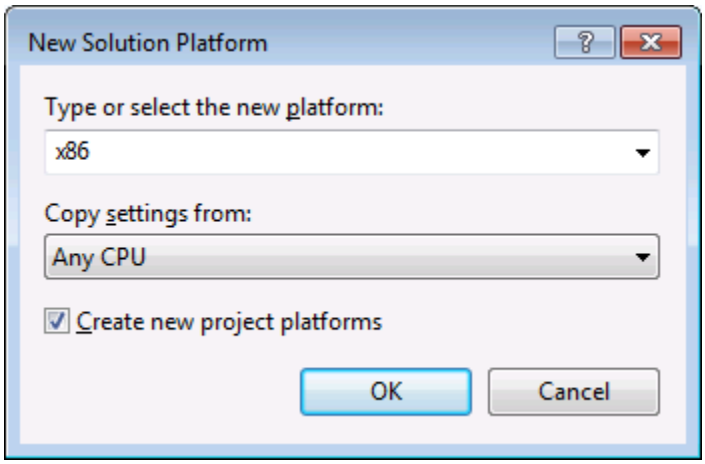
3. ドキュメント概要 ウィンドウ内のActiveX コントロールをクリックし、プロパティ ウィンドウ内の必要とされるプロパティを編集します。例:



ステップ 4: ビルドプラットフォームを設定する

- ビルド(x86、x64)するプラットフォームに一致するビルドプラットフォームの更新を作成します。ビルドの更新の作成の方法は以下のとおりです:

a. Visual Studio 内のソリューションを右クリックし、「 更新マネージャー」を選択します。
b. 「アクティブなソリューションプラットフォーム」から、「新規作成...」を選択し、x86 または x64 更新を選択します（このサンプルでは、**x86**）。



Visual Studio 内でソリューションをビルドして実行する準備が整うと、ターゲットプラットフォーム（x86、x64）に一致するように更新をしようしてビルドすることを忘れないでください。それ以外の場合、ランタイムエラーが起こる可能性があります。

サンプルを実行するには、メイン MDI フレームウィンドウが作成され、この中には空の XML ドキュメントを持つ編集ウィンドウと XMLSpy の XPath ダイアログ ウィンドウが含まれています。「ファイル | 開く」を使用して、XMLSpy サンプルフォルダーから XML ファイルを開きます。ファイルはロードされ表示されます。ドキュメントのロード後、XPath ダイアログの使用を開始します。描きなおしコンテンツを表示するためにフォームの下の角をドラッグする必要があります。

25.4.5.2 HTML

このセクションのコードリストは、アプリケーションレベルとドキュメントレベルでの XMLSpyControl の統合方法を示しています。全てのサンプルのためのソースコードは XMLSpy インストールのフォルダー `<ApplicationFolder>\Examples\ActiveX\HTML` 内にあります。サンプルは、32 ビット アプリケーションとして作動する場合、Internet Explorer 内でのみ作動することができます。

メモ 64 ビット モード上で作動する場合、nternet Explorer 10 または以降は、ActiveX コントロールをロードしません。

25.4.5.2.1 アプリケーションレベルの HTML 統合

このサンプルは、アプリケーションレベルでの HTML ページへの XMLSpy コントロールの簡単な統合について説明しています。統合に関しては、次のセクションで説明されています:

- HTML コード内で XMLSpyControl を開始します。
- ボタンがドキュメントをロードするように実装し、コード生成タスクを自動化します。
- アプリケーションイベントのためのアクションを定義する。

このサンプルのためのコードは、XMLSpy インストール内の次の場所で使用することができます:
`<ApplicationFolder>\Examples\ActiveX\HTML\XMLSpyActiveX_ApplicationLevel.htm`.

25.4.5.2.1.1 コントロールインスタンスを作成する

HTML `Object` タグは、XMLSpyControl のインスタンスを作成するために使用されます。`Classid` は、XMLSpyControl です。幅と高さがウィンドウのサイズを指定します。アプリケーションレベルがデフォルトのため、追加のパラメーターは必要ありません。

```
<OBJECT id="objXMLSpyControl"
        Classid="clsid:a258bba2-3835-4c16-8590-72b44f52c471"
        width="1000"
        height="700"
        VIEWASTEXT>
</OBJECT>
```

25.4.5.2.1.2 デフォルトのドキュメントを開くボタンを追加する

一部のタスクの自動化のシンプルなサンプルとして、ページにボタンを追加します:

```
<input type="button" value="Open Marketing Expenses" onclick="BtnOpenMEFile()">
```

クリックされると、定義済みドキュメントがXMLSpyControl 内で開かれます。異なるインストール上でサンプルを作動するため、XMLSpyControl に対して相対的なファイルを検索するためにメソッドを使用します。

```
<SCRIPT ID=Javahandlers LANGUAGE=javascript>
// ---------------------------------
// open a pre-defined document
function BtnOpenMEFile()
{

              objXMLSpyControl.Open("C:\Documents and Settings\username\My
Documents\Altova\XMLSpy2021\Examples/OrgChart.xml");

}
</SCRIPT>
```

25.4.5.2.1.3 コード生成のためにボタンを追加する

直接アクセスするには、現在のドキュメントを検証するボタンが必要です。メソッドは前のセクションで使用されたメソッドに類似しています。

最初はボタンを与えます:

```
<input type="button" value="Validate" onclick="BtnValidate()">
```

現在のドキュメントを検証するスクリプトを与えます。

```
<SCRIPT ID=Javahandlers LANGUAGE=javascript>
// ---------------------------------------------------------------------
// check validity of current document.
// if validation fails, show validation result in alert box .
function BtnValidate()
{
       // get top-level object of automation interface
       var objApp = objXMLSpyControl.Application;

       // get the active document
```

```
	var objDocument = objApp.ActiveDocument;

	if ( objDocument == null )
		alert( "no active document found" );
	else
	{
		// define as arrays to support their usage as return parameters
		var errorText = new Array(1);
		var errorPos = new Array(1);
		var badData = new Array(1);

		var valid = objDocument.IsValid(errorText, errorPos, badData);

		if (! valid)
		{
			// compose the error description
			var text = errorText;

			// access that XMLData object only if filled in
			if (badData[0] != null)
				text += "(" + badData[0].Name + "/" + badData[0].TextValue
+ ")";


			alert("Validation error[" + errorPos + "]: " + text);
		}
		else
			alert("Docuent is valid"); }
}
</SCRIPT>
```

25.4.5.2.1.4 カスタムイベントへの接続

サンプルは、プリンシパルを表示するためにXMLSpyControl カスタムイベントのための2つのイベントのコールバックを実装します:

```
<!-- ------------------------------------------------------------ -->
<!--  custom event 'OnDocumentOpened" of XMLSpyControl object  -->
<SCRIPT LANGUAGE="javascript">
	function objXMLSpyControl::OnDocumentOpened( objDocument )
	{
	// alert("Document '" + objDocument.Name + "' opened!");
	}
</SCRIPT>

<!-- ------------------------------------------------------------ -->
<!--  custom event 'OnDocumentClosed" of XMLSpyControl object  -->
<SCRIPT LANGUAGE="javascript">
	function objXMLSpyControl::OnDocumentClosed( objDocument )
	{
	// alert("Document '" + objDocument.Name + "' closed!");
	}
</SCRIPT>
```

25.4.5.2.2 ドキュメントレベルの HTML 統合

このサンプルは、ドキュメントレベルで HTML ページに XMLSpy コントロールを統合する方法について説明しています。次のトピックについて説明されています:

- HTML コード内で XMLSpyControl ActiveX コントロールオブジェクトをインスタンス化します。
- XMLSpy ファイルの編集を許可する XMLSpyControlDocument ActiveX コントロールをインスタンス化します。
- XMLSpyControl プロジェクトウィンドウのための XMLSpyControlPlaceHolder をインスタンス化します。
- XMLSpy ヘルパーウィンドウをホストするために XMLSpyControlPlaceHolder をインスタンス化します。
- XMLSpy 検証出力ウィンドウを表示するために XMLSpyControlPlaceHolder ActiveX コントロールをインスタンス化します。
- 頻繁に使用される XMLSpy コマンドのためのシンプルなカスタマーツールバーを作成します。
- XMLSpy の COM 自動化インターフェイスを使用するボタンを追加します。
- コマンドボタンを更新するためのイベントハンドラーを使用します。

このサンプルは、XMLSpy インストールの `<ApplicationFolder>\Examples\ActiveX\HTML\` フォルダー内のファイル `XMLSpyActiveX_ApplicationLevel.htm` 内で使用することがあります。

25.4.5.2.2.1 XMLSpyControl をインスタンス化する

XMLSpyControl HTML `OBJECT` タグは、XMLSpyControl のインスタンスを作成するために使用されます。クラス ID は XMLSpyControl です。ユーザーインターフェイスのためのコントロール間ジェージャーとして使用される幅と高さは、0 に設定されています。統合レベルは `OBJECT` タグ内でパラメーターとして指定されています。

```
<OBJECT id="objXMLSpyXXMLSpyControl"
        Classid="clsid:a258bba2-3835-4c16-8590-72b44f52c471"
        width="0"
        height="0"
        VIEWASTEXT>
   <PARAM NAME="IntegrationLevel" VALUE="1">
</OBJECT>
```

25.4.5.2.2.2 エディターウィンドウの作成

HTML `OBJECT` タグは、編集ウィンドウを埋め込むために使用されます。

```
<OBJECT id="objDoc1"
        Classid="clsid:DFBB0871-DAFE-4502-BB66-08CEB7DF5255"
        width="600"
        height="500"
        VIEWASTEXT>
</OBJECT>
```

25.4.5.2.2.3 プロジェクトウィンドウの作成

HTML OBJECT タグは、XMLSpyControlPlaceHolder ウィンドウを作成するために使用されます。最初の追加カスタムパラメーターは、XMLSpy プロジェクトウィンドウを表示するためのプレースホルダを定義します。2番目のパラメーターは、XMLSpy インストールで提供されたサンプルプロジェクトの1つをロードします（以下の場所で見つけることができます: <yourusername>/MyDocuments フォルダー）。

```
<OBJECT id="objProjectWindow"
        Classid="clsid:FDEC3B04-05F2-427d-988C-F03A85DE53C2"
        width="200"
        height="200"
        VIEWASTEXT>
    <PARAM name="PlaceholderWindowID" value="3">
          <PARAM name="FileName" value="Examples/Examples.spp">
</OBJECT>
```

25.4.5.2.2.4 ヘルパーウィンドウのためにプレースホルダを作成する

HTML OBJECT タグは、異なるXMLSpy ヘルパーウィンドウをホストすることのできるXMLSpyControlPlaceHolder ActiveX コントロールをインスタンス化するために使用されます。最初から、ヘルパーウィンドウは表示されません。サンプルファイルを参照してください。

```
<OBJECT id="objEHWindow"
        Classid="clsid:135DEEF4-6DF0-47c2-8F8C-F145F5F3F672"
        width="200"
        height="200"
        VIEWASTEXT>
       <PARAM name="PlaceholderWindowID" value="0">
</OBJECT>
```

3つのボタンにより表示される実際のウィンドウを切り替えることができます。JavaScript 実行ボタンをクリックすることにより、プロパティ PlaceHolderWindowID を定義されている対応する値に設定します。XMLSpyControlPlaceholderWindow [1479].

```
<input type="button" value="EH - Elements" onclick="BtnHelperWindow(0)">
<input type="button" value="EH - Attributes" onclick="BtnHelperWindow(1)">
<input type="button" value="EH - Entities" onclick="BtnHelperWindow(2)">

<SCRIPT ID="Javahandlers" LANGUAGE="javascript">
// ---------------------------------------------------------------------------
// specify which of the windows shall be shown in the placeholder control.
function BtnHelperWindow(i_ePlaceholderWindowID)
{
       objEHWindow.PlaceholderWindowID = i_ePlaceholderWindowID;
}
</SCRIPT>
```

25.4.5.2.2.5 カスタムツールバーの作成

カスタムツールバーは、XMLSpy コマンドのイメージを持つボタンにより構成されています。コマンド ID 番号は、表示されるボタン要素内で見つけることができます コマンドテーブル [1440]。

```
<button id="btnWellFormed" title="Check Well-formedness" onclick="BtnDoCommand(34049)">
      <img src="..\Images\IDC_CHECK_WELL_FORM.gif" width="16" height="16" />
</button>
<button id="btnValidate" title="Validate" onclick="BtnDoCommand(34174)">
      <img src="..\Images\IDC_VALIDATE.gif" width="16" height="16" />
</button>
```

ボタンをクリックすると、対応するコマンド ID がマネジャーコントロールに送信されます。

```
<SCRIPT ID="Javahandlers" LANGUAGE="javascript">
// ---------------------------------------------------------------
// perform any command specified by cmdID.
// command routing includes application, active document and view.
function BtnDoCommand(cmdID)
{
      objXMLSpyX.Exec( cmdID );
      msgtext.innerText = "Command " + cmdID + " performed.";
}
</SCRIPT>
```

25.4.5.2.2.6 ボタンを追加する

サンプルでは、自動化コードを表示するボタンを追加します。

```
<p>
      <input type="button" value="New File" onclick="BtnNewFile(objDoc1)">
      <input type="button" value="Save File" onclick="BtnSaveFile(objDoc1)">
      <input type="text" title="Path" id="strPath" width="150">
      <input type="button" value="Open OrgChart.xml" onclick="BtnOpenFile(objDoc1,
'OrgChart.xml')">
      <input type="button" value="Open OrgChart.xsd" onclick="BtnOpenFile(objDoc1,
'OrgChart.xsd')">
<p>
```

対応する JavaScript 修正は以下のようになります:

```
<SCRIPT ID="Javahandlers" LANGUAGE="javascript">
// -------------------------------------------------------
// open a document in the specified document control window.
function BtnOpenFile(objDocCtrl, strFileName)
{
      // do not use XMLSpyX.Application.OpenDocument(...) to open a document,
      // since then XMLSpyControl wouldn't know a control window to show
      // the document in. Instead:

      objDocCtrl.Open("C:\Documents and Settings\username\My
Documents\Altova\XMLSpy2021\Examples/" + strFileName);
      objDocCtrl.setActive();

}

// ---------------------------------------------------------------
// open a new empty document in the specified document control window.
function BtnNewFile(objDocCtrl)
{
```

```
	objDocCtrl.Open("");
	objDocCtrl.setActive();
}

// ---------------------------------------------------------------------
// Saves the current file in the specified document control window.
function BtnSaveFile(objDocCtrl)
{
	if(objDocCtrl.Path.length > 0)
		objDocCtrl.SaveDocument();
	else
	{
		if(strPath.value.length > 0)
		{
			objDocCtrl.Path = strPath.value;
			objDocCtrl.Save();
		}
		else
		{
			alert("Please set path for the document first!");
			strPath.focus();
		}
	}

	objDocCtrl.setActive();
}

// ----------------------------------------------------------------------
// check validity of current document.
// if validation fails, show validation result in alert box .
function BtnValidate()
{
	// get top-level object of automation interface
	var objApp = objXMLSpyControl.Application;

	// get the active document
	var objDocument = objApp.ActiveDocument;

	if ( objDocument == null )
		alert( "no active document found" );
	else
	{
		// define as arrays to support their usage as return parameters
		var errorText = new Array(1);
		var errorPos = new Array(1);
		var badData = new Array(1);

		var valid = objDocument.IsValid(errorText, errorPos, badData);

		if (! valid)
		{
			// compose the error description
			var text = errorText;

			// access that XMLData object only if filled in
			if (badData[0] != null)
```

```
                                text += "(" + badData[0].Name + "/" + badData[0].TextValue
+ ")";


                        alert("Validation error[" + errorPos + "]: " + text);
                }
                else
                        alert("Docuent is valid");
        }
}
</SCRIPT>
```

25.4.5.2.2.7 ボタンステータスを更新するためのイベントハンドラーを作成する

コマンドの可用性はマウスクリック、または、キーボード操作により異なる場合があります。XMLSpyControl のカスタムイベント OnUpdateCmdUI は、XMLSpy コマンドに関連するボタンの有効化された/無効化された状態を更新することができます。コマンドの有効化、無効化に関わらず、メソッド XMLSpyControl.QueryStatus[1467] はクエリのために使用されます。

```
<SCRIPT LANGUAGE="javascript">

        function objXMLSpyX::OnUpdateCmdUI()
        {
        if ( document.readyState == "complete" )              // 'complete'
        {
                // update status of buttons
                btnWellFormed.disabled =  ! (objDoc1.QueryStatus(34049) & 0x02);   // not
enabled
                btnValidate.disabled =  ! (objDoc1.QueryStatus(34174) & 0x02);     // not
enabled
        }
        }

        // set activity status of simulated toolbar
</SCRIPT>
```

25.4.5.3 Java

XMLSpy ActiveX コンポーネントにJava コードからアクセスすることができます。Java 統合は下にリストされるライブラリで提供されています。これらのライブラリは、XMLSpy とXMLSpy Integration Package の両方をインストールした後、XMLSpy インストール内のフォルダー <ApplicationFolder>\Examples\JavaAPI で使用することができます(次を参照してください: 必要条件[1413])。

- AltovaAutomation.dll: Altova 自動サーバーのためのJNI ラッパー(XMLSpy の32 ビット インストールの場合)
- AltovaAutomation_x64.dll: Altova 自動サーバーのためのJNI ラッパー(XMLSpy の64 ビット インストールの場合)
- AltovaAutomation.jar: Altova 自動サーバーにアクセスします
- XMLSpyActiveX.jar: XMLSpy ActiveX インターフェイスをラップするJava クラス
- XMLSpyActiveX_JavaDoc.zip: Javadoc インターフェイスのためのヘルプドキュメントを含むJavadoc ファイル

メモ Java ActiveX 統合を使用するには、.dll と.jar ファイルがJava クラス検索パス内に含まれている必要があります。

サンプルJava プロジェクト
サンプルJava プロジェクトは製品のインストールにより提供されます。Java プロジェクト をテストし、使用用途に応じて変更することができます。詳細に関しては、次を参照してください: サンプルJava プロジェクト 1433。

JavaRules にActiveX コントロール名をマッピングする際のルール
ActiveX コントロールのドキュメントに関しては、次を参照してください: オブジェクトレファレンス 1459。オブジェクトの名前変換はJava の場合、他の言語と比較して、若干異なります。具体的には、ActiveX コントロールとJava ラッパー間のマッピングのためのルールは、以下のとおりです:

- *クラスとクラス名*
  XMLSpy ActiveX インターフェイスの全てのコンポーネントのために、コンポーネントの名前を持つJava クラスが存在します。

- *メソッド名*
  Javadoc インターフェイス上のメソッド名は、COM インターフェイス上で使用される名前と同じですが、Java 名前変換に準拠する小文字で開始します。COM プロパティにアクセスするには、プロパティ名をプレフィックスするgetとsetから始まるJava メソッド使用することができます。ふプロパティが書き込みをサポートしない場合、setter メソッドを使用することはできません。サンプル: XMLSpyControl の`IntegrationLevel` プロパティ、Java メソッド `getIntegrationLevel` と `setIntegrationLevel` を使用することができます。

- *列挙*
  ActiveX インターフェイス内で定義される全ての列挙のために、Java 列挙は同じ名前と値を使用して定義されます。

- *イベント とイベントハンドラー*
  イベントをサポートする自動化インターフェイス内の全てのインターフェイスのために、同じ名前と'イベント' を使用したJavadoc インターフェイスを使用することができます。単一のイベントのオーバーロードを簡素化するために、全てのイベントのためのデフォルトの実装を使用するJava クラスが提供されます。このJava クラスの名前は、イベント インターフェイスの名前と '`DefaultHandler`' です。例:
  `XMLSpyControl`:アプリケーションにアクセスするためのJava クラス
  `XMLSpyControlEvents`:XMLSpyControl のためのイベント インターフェイス
  `XMLSpyControlEventsDefaultHandler`:`XMLSpyControlEvents` のためのデフォルトのハンドラー

マッピングルールの例外
上記のルールの例外の一部は、以下のとおりです。これらは下にリストされています:

インターフェイス	Java 名
`XMLSpyControlDocument`, method `New`	`newDocument`
`Document`, method `SetEncoding`	`setFileEncoding`
`AuthenticView`, method `Goto`	`gotoElement`
`AuthenticRange`, method `Goto`	`gotoElement`
`AuthenticRange`, method `Clone`	`cloneRange`

このセクション
このセクションは、Java コードからアクセスすることのできる基本のXMLSpy ActiveX 機能の一部を紹介しています。次のサブセクションに整理されています:

- サンプルJava プロジェクト 1433
- ActiveX コントロールの作成 1435

- コントロール内のデータのロード 1435
- 基本のイベントハンドリング 1436
- メニュー 1436
- UI アップデートイベントハンドリング 1438
- XML ツリーの作成 1438

25.4.5.3.1 サンプル Java プロジェクト

XMLSpy インストールパッケージには、アプリケーションフォルダー: `<ApplicationFolder>\Examples\ActiveX\Java\` の ActiveX サンプルフォルダーにあるサンプル Java プロジェクトが含まれています。

Java サンプルは、Java. を使用して作成された共有されるデスクトップアプリケーション内に XMLSpyControl を統合する方法を示しています。

バッチファイル `BuildAndRun.bat` を使用してコマンドラインからテストすることができ、また、Eclipse 内からサンプルプロジェクトをコンパイルして実行することもできます。これらのプロシージャの使用方法を参照してください。

ファイルリスト

Java サンプルフォルダーには、サンプルプロジェクトを作動するために必要な全てのファイルが含まれています。これらのファイルは下にリストされています:

`.classpath`	Eclipse プロジェクトヘルパーファイル
`.project`	Eclipse プロジェクトファイル
`AltovaAutomation.dll`	Java-COM ブリッジ: DLL パート (32-ビットインストール用)
`AltovaAutomation_x64.dll`	Java-COM ブリッジ: DLL パート (64-ビットインストール用)
`AltovaAutomation.jar`	Java-COM ブリッジ: Java ライブラリパート
`BuildAndRun.bat`	コンパイルするバッチファイル、および、コマンドラインプロンプトからサンプルコードを実行します。Java 仮想マシンがパラメータとして存在する箇所にフォルダーが期待されます。
`XMLSpyActiveX.jar`	XMLSpy ActiveX コントロールの Java クラス
`XMLSpyActiveX_JavaDoc.zip`	Java API のためのヘルプドキュメントを含む Javadoc ファイル
`XMLSpyContainer.java`	Java サンプルソースコード
`XMLSpyContainerEventHandler.java`	Java サンプルソースコード
`XMLTreeDialog.java`	Java サンプルソースコード

サンプルの内容

サンプルplaces one XMLSpy ドキュメント エディターウィンドウ、XMLSpy プロジェクトウィンドウ、XMLSpy XPath ウィンドウと XMLSpy 入力ヘルパーin an AW T フレームウィンドウ。XMLSpy のために定義されているファイルメニューを読み取り、同じ構造を持つ AW T メニューを作成します。このメニューまたはプロジェクトを使用して、ドキュメントエディター内でファイルを開き作業することができます。

サンプルを必要に応じて変更することができます。

コードリスト内で次の特定の機能が説明されています:

- ActiveX コントロールの作成[1435]: 自動サーバーとして考えられているXMLSpy を返します。または、作動中の場合、XMLSpy をアクティブ化します。
- コントロール内のデータのロード[1435]: XMLSpy にインストールされているサンプルドキュメントの1つを検索し開きます。
- 基本のイベントハンドリング[1436]: すべての開かれているドキュメントのビューをテキストビューに変えます。コードも開かれているドキュメント内での反復の方法を示しています。
- メニュー[1436]: アクティブなドキュメントを検証し、メッセージボックス内に結果を表示します。コードは出力パラメータの使用方法について説明しています。
- UI アップデートイベントハンドリング[1438]: XMLSpy イベントの処理方法について説明しています。
- XML ツリーの作成[1438]: XML ツリーの作成とモーダルアクティブ化について説明しています。

サンプルフォルダーへのパスの更新

与えられているサンプルを作動する前に、**XMLSpyContainer.java** ファイルを編集します。具体的には、XMLSpy サンプルファイルが使用中のオペレーティングシステム上で保存されている実際のフォルダーを参照する次のパスを確認します:

```
// Locate samples installed with the product.
final String strExamplesFolder = System.getenv( "USERPROFILE" ) + "\\My Documents\
\Altova\\XMLSpy2021\\XMLSpyExamples\\";
```

コマンドラインからサンプルを実行する

コマンドラインからサンプルを作動します:

1. すべての必要条件が満たされていることを確認してください(次を参照してください: 必要条件[1413]).
2. コマンドプロンプトウィンドウを開き、サンプルJava プロジェクト フォルダーに対して現在のディレクトリを変更し、次を入力します:

```
buildAndRun.bat "<Path-to-the-Java-bin-folder>"
```

3. 「Enter」を押します。

`XMLSpyContainer.java` 内のJava ソースがコンパイルされ、実行されます。

Eclipse 内でサンプルをコンパイルし実行する

サンプルJava プロジェクトをEclipse にインポートする

1. すべての必要条件が満たされていることを確認してください(次を参照してください: 必要条件[1413]).
2. ファイル メニューから、「インポート」をクリックします。
3. 「ワークスペースで既存のプロジェクトを開く」を選択し、次の場所にあるEclipse プロジェクトファイルを参照します: <ApplicationFolder>¥Examples¥ActiveX¥Java¥。このフォルダーでは書き込みのアクセスがないため、インポートダイアログボックス上の「ワークプレースにプロジェクトをコピーする」チェックボックスを選択することが奨励されます。

サンプルアプリケーションを作動するには、Package Explorer 内のプロジェクトを右クリックし、コマンド「実行 | Java アプリケーション」を選択します。

コード内のコメントを使用するとJava API クラスのためのヘルプを使用することができ、とEclipse 内でJavadoc ビューを有効化するには、メニューコマンド「Window | ビューの表示 | JavaDoc」を選択してください。

25.4.5.3.2 ActiveX コントロールの作成

下のコードリストは、ActiveX コントロールの作成方法を示しています。コンストラクターは、Java ラッパーオブジェクトを作成します。カンバスにより派生したオブジェクトをパネル、または、フレームに追加すると、ラップされた ActiveX の生成がトリガーされます。

```
01  /**
02  * XMLSpy manager control - always needed
03  */
04    public static XMLSpyControl         xmlSpyControl = null;
05
06  /**
07  * XMLSpy document editing control
08  */
09    public static XMLSpyControlDocument     xmlSpyDocument = null;
10
11  /**
12  * Tool windows - XMLSpy place-holder controls
13  */
14    private static XMLSpyControlPlaceHolder   xmlSpyProjectToolWindow = null;
15    private static XMLSpyControlPlaceHolder   xmlSpyXpathToolWindow = null;
16    private static XMLSpyControlPlaceHolder   xmlSpyEHAttributeToolWindow = null;
17
18  // Create the XMLSpy ActiveX control; the parameter determines that we want
19  // to place document controls and place-holder controls individually.
20  // It gives us full control over the menu, as well.
21     xmlSpyControl = new XMLSpyControl(
       ICActiveXIntegrationLevel.ICActiveXIntegrationOnDocumentLevel.getValue() );
22     xmlSpyDocument = new XMLSpyControlDocument();
23     xmlSpyDocument.setPreferredSize( new Dimension ( 640, 480 ) );
24
25  // Create a project window and open the sample project in it
26     xmlSpyProjectToolWindow = new XMLSpyControlPlaceHolder(
27     XMLSpyControlPlaceholderWindow.XMLSpyControlProjectWindowToolWnd.getValue() );
28     xmlSpyProjectToolWindow.setPreferredSize( new Dimension( 200, 200 ) );
29     xmlSpyXpathToolWindow = new XMLSpyControlPlaceHolder(
       XMLSpyControlPlaceholderWindow.XMLSpyControlXPathDialogToolWnd.getValue() );
30     xmlSpyEHAttributeToolWindow = new XMLSpyControlPlaceHolder(
       XMLSpyControlPlaceholderWindow.XMLSpyControlEntryHelperTopToolWnd.getValue() );
31
32     frame.add( xmlSpyControl, BorderLayout.NORTH );
33     frame.add( xmlSpyDocument, BorderLayout.CENTER );
34     southPanel.add( xmlSpyProjectToolWindow );
35     southPanel.add( xmlSpyXpathToolWindow );
36     southPanel.add( xmlSpyEHAttributeToolWindow );
```

25.4.5.3.3 コントロール内のデータのロード

下のコードリストは、ActiveX コントロール内にデータをロードを表示する方法を表示しています。

```
1    // Locate samples installed with the product.
2       final String strExamplesFolder = System.getenv( "USERPROFILE" ) +
        "\\My Documents\\Altova\\XMLSpy2021\\Examples\\";
3       xmlSpyProjectToolWindow.openProject( strExamplesFolder  + "Examples.spp" );
```

25.4.5.3.4 基本のイベントハンドリング

下のコードリストは、基本のイベントの処理方法について表示しています。XMLSpyControl の open メソッド、または、メニューまたはプロジェクトツリーを使用してファイルを開こうとすると、onOpenedOrFocused イベントは、アタッチ済みのイベントハンドラーに送信されます。このイベントのための基本の処理により、XMLSpyDocumentControl の open メソッドを呼び出すことによりファイルを開きます。

```
01       // Open the PXF file when button is pressed
02       btnOpenPxf.addActionListener( new ActionListener() {
03         public void actionPerformed(ActionEvent e) {
04           try {
05             xmlSpyControl.open( strExamplesFolder + "OrgChart.pxf" );
06           } catch (AutomationException e1) {
07             e1.printStackTrace();
08           }
09         }
10       } );
11       public void onOpenedOrFocused( String i_strFileName, boolean i_bOpenWithThisControl,
boolean i_bFileAlreadyOpened ) throws AutomationException
12   {
13     // Handle the New/Open events coming from the Project tree or from the menus
14     if ( !i_bFileAlreadyOpened )
15     {
16       // This is basically an SDI interface, so open the file in the already existing document
control
17       try {
18         XMLSpyContainer.xmlSpyDocument.open( i_strFileName );
19         XMLSpyContainer.xmlSpyDocument.requestFocusInWindow();
20       } catch (Exception e) {
21         e.printStackTrace();
22       }
23     }
24   }
```

25.4.5.3.5 メニュー

下のコードリストは、メニューアイテムの作成方法を表示しています。 各 XMLSpyCommand オブジェクトは、コマンドのID へのセットされている ActionCommand を持つ、対応する MenuItem オブジェクト、を取得します。全てのメニューアイテムにより生成されているアクションは、(閉じられたメカニズムの)を再度解釈するなどの)特定の処理を行う同じ関数により処理されます。または、exec メソッドを呼び出すことにより XMLSpyControl オブジェクトに実行を委任することができます。メニューの作成中にフィルされる menuMap オブジェクトは、後で使用されます(次のセクションを参照してください: UI アップデートイベントハンドリング 1438).

```
01  // Load the file menu when the button is pressed
02       btnMenu.addActionListener( new ActionListener() {
03         public void actionPerformed(ActionEvent e) {
04           try {
05             // Create the menubar that will be attached to the frame
06             MenuBar mb = new MenuBar();
07             // Load the main menu's first item - the File menu
08             XMLSpyCommand xmlSpyMenu = xmlSpyControl.getMainMenu().getSubCommands().getItem( 0 );
09             // Create Java menu items from the commands objects
10             Menu fileMenu = new Menu();
11             handlerobject.fillMenu( fileMenu, xmlSpyMenu.getSubCommands() );
12             fileMenu.setLabel( xmlSpyMenu.getLabel().replace( "&", "" ) );
13             mb.add( fileMenu );
14             frame.setMenuBar( mb );
```

```
15              frame.validate();
16            } catch (AutomationException e1) {
17              e1.printStackTrace();
18            }
19            // Disable the button when the action has been performed
20            ((AbstractButton) e.getSource()).setEnabled( false );
21          }
22        } ) ;
23 /** * Populates a menu with the commands and submenus contained in an XMLSpyCommands object  */
24        public void fillMenu(Menu newMenu, XMLSpyCommands xmlSpyMenu) throws AutomationException
25    {
26      // For each command/submenu in the xmlSpyMenu
27      for ( int i = 0 ; i < xmlSpyMenu.getCount() ; ++i  )
28      {
29        XMLSpyCommand xmlSpyCommand = xmlSpyMenu.getItem( i );
30        if ( xmlSpyCommand.getIsSeparator() )
31          newMenu.addSeparator();
32        else
33        {
34          XMLSpyCommands subCommands = xmlSpyCommand.getSubCommands();
35          // Is it a command (leaf), or a submenu?
36          if ( subcommands.isNull() || subCommands.getCount() == 0 )
37          {
38            // Command -> add it to the menu, set its ActionCommand to its ID and store it in the
menuMap
39            MenuItem mi = new MenuItem( xmlSpyCommand.getLabel().replace( "&", "" ) );
40            mi.setActionCommand( "" + xmlSpyCommand.getID() );
41            mi.addActionListener( this );
42            newMenu.add( mi );
43            menuMap.put( xmlSpyCommand.getID(), mi );
44          }
45          else
46          {
47            // Submenu -> create submenu and repeat recursively
48            Menu newSubMenu = new Menu();
49            fillMenu( newSubMenu, subCommands );
50            newSubMenu.setLabel( xmlSpyCommand.getLabel().replace( "&", "" ) );
51            newMenu.add( newSubMenu );
52          }
53        }
54      }
55    }
56
57     /**
58     * Action handler for the menu items
59     * Called when the user selects a menu item; the item's action command corresponds to the
command table for XMLSpy
60     */
61    public void actionPerformed( ActionEvent e )
62    {
63      try
64      {
65        int iCmd = Integer.parseInt( e.getActionCommand() );
66        // Handle explicitly the Close commands
67        switch ( iCmd )
68        {
69          case 57602:       // Close
70          case 34050:       // Close All
71            XMLSpyContainer.initXmlSpyDocument();
72            break;
73          default:
74            XMLSpyContainer.xmlSpyControl.exec( iCmd );
75            break;
76        }
77      }
```

```
78      catch ( Exception ex )
79      {
80        ex.printStackTrace();
81      }
82
83    }
```

25.4.5.3.6 UI アップデートイベントハンドリング

下のコードリストは、UI-Update イベントハンドラーの作成方法を表示しています。

```
01 /**
02    * Call-back from the XMLSpyControl.
03    * Called to enable/disable commands
04    */
05   @Override
06   public void onUpdateCmdUI() throws AutomationException
07   {
08     // A command should be enabled if the result of queryStatus contains the Supported (1) and
Enabled (2) flags
09     for ( java.util.Map.Entry<Integer, MenuItem> pair : menuMap.entrySet() )
10       pair.getValue().setEnabled( XMLSpyContainer.xmlSpyControl.queryStatus( pair.getKey() ) >
2 );
11   }
12    /**
13    * Call-back from the XMLSpyControl.
14    * Usually called while enabling/disabling commands due to UI updates
15    */
16   @Override
17   public boolean onIsActiveEditor( String i_strFilePath ) throws AutomationException
18   {
19     try {
20       return
XMLSpyContainer.xmlSpyDocument.getDocument().getFullName().equalsIgnoreCase( i_strFilePath );
21     } catch ( Exception e ) {
22       return false;
23     }
24   }
```

25.4.5.3.7 XML ツリーの作成

下のリストは、XML データオブジェクトをツリー内にノードとしてロードします。

```
01 // access required XMLSpy Java-COM classes
02 import com.altova.automation.XMLSpy.XMLData;
03
04 // access AWT and Swing components
05 import java.awt.*;
06 import javax.swing.*;
07 import javax.swing.tree.*;
08
09 /**
10  * A simple example of a tree control loading the structure from an XMLData object.
11  * The class receives an XMLData object, loads its nodes in a JTree, and prepares
12  * for modal activation.
13  *
```

```
14  * Feel free to modify and extend this sample.
15  *
16  * @author Altova GmbH
17  */
18 class XMLTreeDialog extends JDialog
19 {
20   /**
21    * The tree control
22    */
23   private JTree myTree;
24
25   /**
26    * Root node of the tree control
27    */
28   private DefaultMutableTreeNode top ;
29
30   /**
31    * Constructor that prepares the modal dialog containing the filled tree control
32    * @param xml   The data to be displayed in the tree
33    * @param parent  Parent frame
34    */
35   public XMLTreeDialog( XMLData xml, Frame parent )
36   {
37     // Construct the modal dialog
38     super( parent, "XML tree", true );
39     // Arrange controls in the dialog
40     top = new DefaultMutableTreeNode("root");
41     myTree = new JTree(top);
42     setContentPane( new JScrollPane( myTree ) );
43     // Build up the tree
44     fillTree( top, xml );
45     myTree.expandRow( 0 );
46   }
47
48   /**
49    * Loads the nodes of an XML element under a given tree node
50    * @param node  Target tree node
51    * @param elem  Source XML element
52    */
53   private void fillTree( DefaultMutableTreeNode node, XMLData elem)
54   {
55     try
56     {
57       // There are several ways to iterate through child elements: either using the
getFirstChild/getNextChild,
58       // or by incrementing an index up to countChildren and calling getChild [as shown below].
59       // If you only want to get children of one kind, you should use
countChildrenKind/getChildKind,
60       // or provide a kind to the getFirstChild before iterating with the getNextChild.
61       int nSize = elem.countChildren() ;
62       for ( int i = 0 ; i < nSize ; ++i)
63       {
64         // Create a new tree node for each child element, and continue recursively
65         XMLData newElem = elem.getChild(i) ;
66         DefaultMutableTreeNode newNode = new DefaultMutableTreeNode( newElem.getName() ) ;
67         node.add( newNode ) ;
68         fillTree( newNode, newElem ) ;
69       }
70     }
71     catch (Exception e)
72     {
73       e.printStackTrace();
74     }
75   }
```

```
76
77 }
```

25.4.6 コマンド レファレンス

このセクションは、XMLSpy 内で使用することのできる全てのメニューコマンドの名前と識別子をリストしています。各サブセクションは XMLSpy. コマンドテーブルの対応するトップレベルメニューからのコマンドが以下のとおりに整理されています:

- 「メニューアイテム」列は、コマンドのメニューテキストを XMLSpy で表示される順番に表示します。コマンドの機能を簡単に識別することができます。
- 「コマンド名」列は、XMLSpy インストールディレクトリの **ActiveX\Images** フォルダーから同じ名前を持つアイコンを取得するために使用される文字列を指定します。
- 「ID」列は、このコマンドを実行またはクエリするメソッドの引数として与えられる列の数値識別子を表示します。

コマンドを実行するには、XMLSpyControl.Exec 1466 または XMLSpyControlDocument.Exec 1472 メソッドを利用します。コマンドのステータスをクエリするには、XMLSpyControl.QueryStatus 1467 または XMLSpyControlDocument.QueryStatus 1473 メソッドを使用します。

インストールされた XMLSpy のエディションにより、コマンドの一部がサポートされない場合があります。

25.4.6.1 「ファイル」メニュー

「ファイル」メニューには以下のコマンドが含まれます:

メニューアイテム	コマンド名	ID
新規作成...	ID_FILE_NEW	57600
開く...	ID_FILE_OPEN	57601
再ロード	IDC_FILE_RELOAD	34065
エンコード	IDC_ENCODING	34061
閉じる	ID_FILE_CLOSE	57602
全て閉じる	IDC_CLOSE_ALL	34050
アクティブなファイル以外閉じる	IDC_CLOSE_OTHERS	34271
保存	ID_FILE_SAVE	57603
名前を付けて保存	ID_FILE_SAVE_AS	57604
全て保存する	ID_FILE_SAVE_ALL	34208
電子メールで送信する	ID_FILE_SEND_MAIL	57612
印刷	ID_FILE_PRINT	57607
印刷プレビュー	IDC_PRINT_PREVIEW	34104

メニューアイテム	コマンド名	ID
印刷設定	ID_FILE_PRINT_SETUP	57606
最近使ったファイル	ID_FILE_MRU_FILE1	57616
終了	ID_APP_EXIT	57665

25.4.6.2 「編集」メニュー

「編集」メニューには以下のコマンドが含まれます:

メニューアイテム	コマンド名	ID
元に戻す	ID_EDIT_UNDO	57643
やり直し	ID_EDIT_REDO	57644
切り取り	ID_EDIT_CUT	57635
コピー	ID_EDIT_COPY	57634
貼り付け	ID_EDIT_PASTE	57637
削除	ID_EDIT_CLEAR	57632
XML テキストとしてコピー	IDC_COPY_AS_XML_TEXT	33443
構成されたテキストとしてコピー	IDC_COPY_AS_STRUCTURED_TEXT	33442
XPath のコピー	IDC_COPY_XPATH	33444
XPointer のコピー	IDC_COPY_XPOINTER	33445
ファイル パス...	IDC_EDIT_INSERT_PATH_STRING	34013
XInclude...	IDC_EDIT_INSERT_XINCLUDE_STRING	34017
エンコードされた外部ファイル	IDC_EDIT_INSERT_ENCODED_BINARY_STRING	34273
整形出力	IDC_PRETTY_PRINT	34101
空白スペースを取り去る	IDC_STRIP_WHITESPACES	34296
全て選択	ID_EDIT_SELECT_ALL	57642
検索	ID_EDIT_FIND	57636
次を検索	ID_EDIT_REPEAT	57640
置換	ID_EDIT_REPLACE	57641
ファイル名の検索...	IDC_FIND_IN_FILES	34000

メニューアイテム	コマンド名	ID
ブックマークの挿入/削除	IDC_TOGGLE_BOOKMARK	34162
全てのブックマークの削除	IDC_REMOVEALLBOOKMARKS	34132
次のブックマーク移動	IDC_GOTONEXTBOOKMARK	34070
前のブックマークへ移動	IDC_GOTOPREVBOOKMARK	34071
コメント イン/アウト	IDC_TOGGLE_XML_COMMENT	34029

25.4.6.3 「プロジェクト」メニュー

「プロジェクト」メニューには以下のコマンドが含まれます:

メニューアイテム	コマンド名	ID
新規のプロジェクト	IDC_ICPROJECTGUI_NEW	37200
プロジェクトを開く	IDC_ICPROJECTGUI_OPEN	37201
プロジェクトを再ロードする	IDC_ICPROJECTGUI_RELOAD	37202
プロジェクトを閉じる	IDC_ICPROJECTGUI_CLOSE	37203
プロジェクトを保存する	IDC_ICPROJECTGUI_SAVE	37204
プロジェクトに名前を付けて保存する	IDC_ICPROJECTGUI_SAVE_AS	37207
ソース管理を有効化する	ID_SCC_ENABLE	38602
プロジェクトにファイルを追加する	IDC_ICPROJECTGUI_ADD_FILES_TO_PROJECT	37205
グローバルリソースをプロジェクトに追加する	IDC_ICPROJECTGUI_ADD_GLOBAL_RESOURCE_TO_PROJECT	37239
URL をプロジェクトに追加する	IDC_ICPROJECTGUI_ADD_URL_TO_PROJECT	37206
アクティブなファイルをプロジェクトに追加する	IDC_ICPROJECTGUI_ADD_ACTIVE_FILE_TO_PROJECT	37208
アクティブならびに関係するファイルをプロジェクトに追加する	IDC_ICPROJECTGUI_ADD_ACTIVE_AND_RELATED_FILES_TO_PROJECT	37209
プロジェクト フォルダーをプロジェクトに追加する	IDC_ICPROJECTGUI_ADD_FOLDER_TO_PROJECT	37210
外部フォルダーをプロジェクトに追加する	IDC_ICPROJECTGUI_ADD_EXT_FOLDER_TO_PROJECT	37211

(C) 2015-2021 Altova GmbH

メニューアイテム	コマンド名	ID
外部 Web フォルダーをプロジェクトに追加する	IDC_ICPROJECTGUI_ADD_EXT_URL_FOLDER_TO_PROJECT	37212
スクリプト設定...	IDC_PROJECT_SCRIPT_SETTINGS	34136
プロパティ...	IDC_ICPROJECTGUI_PROJECT_PROPERTIES	37223
最近使ったプロジェクト	IDC_ICPROJECTGUI_RECENT	37224

25.4.6.4 「XML」メニュー

「XML」メニューには以下のコマンドが含まれます:

メニューアイテム	コマンド名	ID
属性	IDC_INSERT_ATTRIBUTE	33449
要素	IDC_INSERT_STRUCT	33459
テキスト	IDC_INSERT_TEXT	33460
CDATA	IDC_INSERT_CDATA	33450
コメント	IDC_INSERT_COMMENT	33451
XML	IDC_INSERT_XML	33461
処理命令	IDC_INSERT_PI	33458
XInclude...	IDC_INSERT_XINCLUDE	34019
DOCTYPE	IDC_INSERT_DEF_DOCTYPE	33453
ExternalID	IDC_INSERT_DEF_EXTERNAL_ID	33456
ELEMENT	IDC_INSERT_DEF_ELEMENT	33454
ATTLIST	IDC_INSERT_DEF_ATTLIST	33452
ENTITY	IDC_INSERT_DEF_ENTITY	33455
NOTATION	IDC_INSERT_DEF_NOTATION	33457
エンコードされた外部ファイル	IDC_INSERT_ENCODED_BINARY	34274
属性	IDC_APPEND_ATTRIBUTE	33415
要素	IDC_APPEND_STRUCT	33425
テキスト	IDC_APPEND_TEXT	33426

メニューアイテム	コマンド名	ID
CDATA	IDC_APPEND_CDATA	33416
コメント	IDC_APPEND_COMMENT	33417
XML	IDC_APPEND_XML	33427
処理命令	IDC_APPEND_PI	33424
XInclude...	IDC_APPEND_XINCLUDE	34026
DOCTYPE	IDC_APPEND_DEF_DOCTYPE	33419
ExternalID	IDC_APPEND_DEF_EXTERNAL_ID	33422
ELEMENT	IDC_APPEND_DEF_ELEMENT	33420
ATTLIST	IDC_APPEND_DEF_ATTLIST	33418
ENTITY	IDC_APPEND_DEF_ENTITY	33421
NOTATION	IDC_APPEND_DEF_NOTATION	33423
エンコードされた外部ファイル	IDC_APPEND_ENCODED_BINARY	34276
属性	IDC_ADD_CHILD_ATTRIBUTE	33402
要素	IDC_ADD_CHILD_STRUCT	33412
テキスト	IDC_ADD_CHILD_TEXT	33413
CDATA	IDC_ADD_CHILD_CDATA	33403
コメント	IDC_ADD_CHILD_COMMENT	33404
XML	IDC_ADD_CHILD_XML	33414
処理命令	IDC_ADD_CHILD_PI	33411
XInclude...	IDC_ADD_CHILD_XINCLUDE	34027
DOCTYPE	IDC_ADD_CHILD_DEF_DOCTYPE	33406
ExternalID	IDC_ADD_CHILD_DEF_EXTERNAL_ID	33409
ELEMENT	IDC_ADD_CHILD_DEF_ELEMENT	33407
ATTLIST	IDC_ADD_CHILD_DEF_ATTLIST	33405
ENTITY	IDC_ADD_CHILD_DEF_ENTITY	33408
NOTATION	IDC_ADD_CHILD_DEF_NOTATION	33410
エンコードされた外部ファイル	IDC_ADD_CHILD_ENCODED_BINARY	34277
属性	IDC_CONVERT_TO_ATTRIBUTE	33429

(C) 2015-2021 Altova GmbH

メニューアイテム	コマンド名	ID
要素	IDC_CONVERT_TO_STRUCT	33439
テキスト	IDC_CONVERT_TO_TEXT	33440
CDATA	IDC_CONVERT_TO_CDATA	33430
コメント	IDC_CONVERT_TO_COMMENT	33431
XML	IDC_CONVERT_TO_XML	33441
処理命令	IDC_CONVERT_TO_PI	33438
DOCTYPE	IDC_CONVERT_TO_DEF_DOCTYPE	33433
ExternalID	IDC_CONVERT_TO_DEF_EXTERNAL_ID	33436
ELEMENT	IDC_CONVERT_TO_DEF_ELEMENT	33434
ATTLIST	IDC_CONVERT_TO_DEF_ATTLIST	33432
ENTITY	IDC_CONVERT_TO_DEF_ENTITY	33435
NOTATION	IDC_CONVERT_TO_DEF_NOTATION	33437
テーブルとして表示	IDC_GRID_VIEW_AS_TABLE	34075
行の挿入	IDC_TABLE_INSERT_ROW	34158
行の追加	IDC_TABLE_APPEND_ROW	34157
昇順の並べ替えと	IDC_TABLE_SORT_ASC	33464
降順の並べ替え	IDC_TABLE_SORT_DESC	33465
左に移動	IDC_MOVE_LEFT	34091
右に移動	IDC_MOVE_RIGHT	34092
要素の中へ	IDC_ENCLOSE_IN_ELEMENT	33446
XPath の検証...	IDC_EVALUATE_XPATH	34007
整形式のチェック	IDC_CHECK_WELL_FORM	34049
XML の検証	IDC_VALIDATE	32954
サーバー上でXML の検証(ハイパフォーマンス)	IDC_VALIDATE_RAPTOR	34309
入力ヘルパーの更新	IDC_UPDATE_ELEMENT_CHOICE	34173
名前スペースプレフィックス...	IDC_NAMESPACE	33462
XML 署名の作成...	IDC_XML_SIGNATURE_CREATE	34280
XML 署名の検証...	IDC_XML_SIGNATURE_VERIFY	34281

25.4.6.5 「DTD/スキーマ」メニュー

「DTD/スキーマ」メニューには以下のコマンドが含まれます:

メニューアイテム	コマンド名	ID
DTD の割り当て...	IDC_ASSIGN_DTD	34032
スキーマの割り当て...	IDC_ASSIGN_SCHEMA	34033
他のDTD を取り込み...	IDC_INCLUDE_DTD	34084
DTD へ移動	IDC_GOTO_DTD	34072
スキーマへ移動	IDC_GOTO_SCHEMA	34074
定義へ移動	IDC_GOTO_DEFINITION	33447
DTD/スキーマの生成...	IDC_GENERATE_DTD_SCHEMA	34068
DTD のフラット化...	IDC_FLATTEN_DTD	34301
DTD をスキーマに変換...	IDC_CONVERT_DTD_TO_SCHEMA	34299
スキーマのフラット化...	IDC_FLATTEN_SCHEMA	34302
スキーマをDTD に変換...	IDC_CONVERT_SCHEMA_TO_DTD	34300
UML へ変換...	IDC_CONVERT_SCHEMA_TO_UML	34008
MapForce でDB、Excel、EDI からXML を生成...	IDC_DTD_OPENIN_MAPFORCE	34056
StyleVision でHTML/PDF/W ord 出力をデザイン...	IDC_DTD_OPENIN_STYLEVISION	34057
サンプルXML/JSON ファイルの生成...	IDC_GENERATE_XML_FROM_SCHEMA	34069
プログラムコードの生成...	IDC_GENERATE_CODE_FROM_SCHEMA	34067
メモリキャッシュの破棄	IDC_FLUSH_CACHED_FILES	34066

25.4.6.6 「スキーマ設計」メニュー

「スキーマ設計」メニューには以下のコマンドが含まれます:

メニューアイテム	コマンド名	ID
スキーマ設定...	IDC_SCHEMA_NAMESPACES	33571
図の保存...	IDC_SCHEMA_SAVE_DIAGRAM	33581
ドキュメントの生成	IDC_SCHEMA_DOCUMENTATION	34146

(C) 2015-2021 Altova GmbH

メニューアイテム	コマンド名	ID
表示設定...	IDC_SCHEMA_VIEW_CONFIG	33593
ズーム...	IDC_SCHEMA_ZOOM	34150
全てのグローバルを表示	IDC_SCHEMA_MODE_GLOBALS	34147
図を表示	IDC_SCHEMA_MODE_DIAGRAM	33570
Oracle スキーマ拡張子を有効にする	IDC_SCHEMA_ORACLE_EXTENSIONS	33577
Oracle スキーマ設定...	IDC_SCHEMA_ORACLE_SCHEMA_SETTINGS	33578
SQL Server スキーマ拡張子を有効にする	IDC_SCHEMA_SQLSERVER_EXTENSIONS	33588
名前付きスキーマリレーションシップ...	IDC_SCHEMA_SQLSERVER_GLOBAL_RELATIONSHIPS	33589
名前なしスキーマリレーションシップ...	IDC_SCHEMA_SQLSERVER_LOCAL_RELATIONSHIPS	33590
SchemaAgent Server に接続する...	IDC_SCHEMA_SCHEMAAGENT_SERVER_CONNECT	33582
SchemaAgent Server から切断	IDC_SCHEMA_SCHEMAAGENT_SERVER_DISCONNECT	33583
ファイルのみ	IDC_SCHEMAAGENT_SHOW_FILE_ONLY	33504
直接関連付けられているスキーマファイルも表示	IDC_SCHEMAAGENT_SHOW_WITH_DIRECTLY_REFERENCED_SCHEMAS	33608
直接参照しているスキーマファイルも表示	IDC_SCHEMAAGENT_SHOW_WITH_DIRECTLY_REFERENCING_SCHEMAS	33602
直接関連しているスキーマファイルも表示	IDC_SCHEMAAGENT_SHOW_WITH_DIRECTLY_RELATED_SCHEMAS	33613
SchemaAgent 検証...	IDC_SCHEMA_EXTVALID_MENU	33539
スキーマサブセットの作成...	IDC_SCHEMA_CREATE_SUBSET	33650
スキーマのフラット化...	IDC_SCHEMA_FLATTEN	33651

25.4.6.7 「XSL/XQuery」メニュー

「XSL/XQuery」メニューには以下のコマンドが含まれます:

メニューアイテム	コマンド名	ID
XSL 変換	IDC_TRANSFORM_XSL	33006

メニューアイテム	コマンド名	ID
XSL スピードオプティマイザー	IDC_TRANSFORM_XSLPBO	34306
XSL-FO 変換	IDC_TRANSFORM_XSLFO	33007
XSL パラメーター / XQuery 変数...	IDC_TRANSFORM_XSL_PARAMS	33008
XQuery/Update 実行	IDC_TRANSFORM_XQUERY	34170
バックマッピングの有効化	IDC_ENABLE_BACKMAPPING	34364
XSLT/XQuery プロファイリングの有効化....	IDC_PROFILING_OPTIONS	34105
XSL の割り当て...	IDC_ASSIGN_XSL	33001
XSL-FO の割り当て...	IDC_ASSIGN_XSLFO	33002
サンプルXML ファイルの割り当て...	IDC_ASSIGN_SAMPLE_XML	33000
XSL へ移動	IDC_GOTO_XSL	33004
デバッガーの開始 / 移動	ID_PROCESS_XSL	34212
デバッガーの停止	ID_XSLT_DEBUGGER_STOP	33017
デバッガーの再起動	ID_XSLT_DEBUGGER_RESTART	33013
デバッグセッションの終了	ID_XSLT_DEBUGGER_END_SESSION	33011
ステップイン	ID_XSLT_DEBUGGER_STEP	33014
ステップアウト	ID_XSLT_DEBUGGER_STEP_OUT	33015
ステップオーバー	ID_XSLT_DEBUGGER_STEP_OVER	33016
現在の実行ノードを表示	ID_XSLT_DEBUGGER_GO_TO_CURRENT_EXECUTION_NODES	33012
ブレークポイントの挿入/削除	IDC_TOGGLE_BREAKPOINT	34246
トレースポイントの挿入/削除	IDC_TOGGLE_TRACEPOINT	34248
ブレークポイントの有効化/無効化	IDC_ENABLE_BREAKPOINT	34245
トレースポイントの有効化/無効化	IDC_ENABLE_TRACEPOINT	34247
ブレークポイント/トレースポイント...	ID_XSLTDEBUGGER_BREAKPOINTS	33009
コールスタック	ID_XSL_DEBUGWINDOWS_CALLSTACK	34238
XPath ウォッチ	ID_XSL_DEBUGWINDOWS_WATCH	34244
コンテキスト	ID_XSL_DEBUGWINDOWS_CONTEXT	34239
変数	ID_XSL_DEBUGWINDOWS_VARIABLE	34243

メニューアイテム	コマンド名	ID
メッセージ	ID_XSL_DEBUGWINDOWS_MESSAGES	34240
テンプレート	ID_XSL_DEBUGWINDOWS_TEMPLATES	34241
情報	ID_XSLXQUERY_DEBUGWINDOWS_INFO	34237
トレース	ID_XSL_DEBUGWINDOWS_TRACES	34242
デバッグ設定...	ID_XSLTDEBUGGER_SETTINGS	33010

25.4.6.8 「Authentic」メニュー

「Authentic」メニューには以下のコマンドが含まれます:

メニューアイテム	コマンド名	ID
新規ドキュメント...	IDC_AUTHENTIC_NEW_FILE	34036
データベースデータの編集...	IDC_AUTHENTIC_EDIT_DB	34035
StyleVision スタイルシートの割り当て...	IDC_ASSIGN_SPS	34034
StyleVision スタイルシートの編集	IDC_EDIT_SPS	34060
XML データの新たな行を選択し編集	IDC_CHANGE_WORKING_DB_XML_CELL	32861
XML 署名...	IDC_AUTHENTICGUI_XMLSIGNATURE	32862
XML エンティティを定義	IDC_DEFINE_ENTITIES	32805
マークアップを隠す	IDC_MARKUP_HIDE	32855
小さなマークアップの表示	IDC_MARKUP_SMALL	32858
大きなマークアップの表示	IDC_MARKUP_LARGE	32856
混合マークアップの表示	IDC_MARKUP_MIXED	32857
太字	IDC_AUTHENTICGUI_RICHEDIT_TOGGLEBOLD	32813
イタリック	IDC_AUTHENTICGUI_RICHEDIT_TOGGLEITALIC	32814
下線	IDC_AUTHENTICGUI_RICHEDIT_TOGGLEUNDERLINE	32815
取り消し線	IDC_AUTHENTICGUI_RICHEDIT_TOGGLESTRIKETHROUGH	32816
前景色	IDC_AUTHENTICGUI_RICHEDIT_COLOR_FOREGROUND	32824

メニューアイテム	コマンド名	ID
背景色	IDC_AUTHENTICGUI_RICHEDIT_COLOR_BACKGROUND	32830
左寄せ	IDC_AUTHENTICGUI_RICHEDIT_ALIGN_LEFT	32818
中央揃え	IDC_AUTHENTICGUI_RICHEDIT_ALIGN_CENTER	32819
右寄せ	IDC_AUTHENTICGUI_RICHEDIT_ALIGN_RIGHT	32820
行の追加	IDC_ROW_APPEND	32806
行の挿入	IDC_ROW_INSERT	32809
行の複製	IDC_ROW_DUPLICATE	32808
行を上に移動	IDC_ROW_MOVE_UP	32811
行を下に移動	IDC_ROW_MOVE_DOWN	32810
行の削除	IDC_ROW_DELETE	32807
HTML ドキュメントの生成	IDC_PXF_GENERATE_HTML	34283
RTF ドキュメントの生成	IDC_PXF_GENERATE_RTF	34284
PDF ドキュメントの生成	IDC_PXF_GENERATE_PDF	34285
W ord 2007+ ドキュメントの生成	IDC_PXF_GENERATE_DOCX	34286
信頼できる場所	IDC_TRUSTED_LOCATIONS	34288

25.4.6.9 「データベース」メニュー

「DB」メニューには以下のコマンドが含まれます:

メニューアイテム	コマンド名	ID
データベースのクエリ	IDC_QUERYDATABASE	34012
XML スキーマの管理	IDC_DB_MANAGESCHEMAS	34014
XML スキーマの割り当て	IDC_DB_CHOOSEVALIDATIONSCHEMA	34016
XML スキーマの管理	IDC_DB_MANAGESCHEMAS	34014
XML スキーマの管理	IDC_DB_MANAGESCHEMAS	34014
Oracle XML ドキュメントの参照	ID_CONVERT_ORACLEXMLDB_BROWSE	34205

25.4.6.10 「変換」メニュー

「変換」メニューには以下のコマンドが含まれます:

メニューアイテム	コマンド名	ID
テキストファイルをインポート...	IDC_IMPORT_TEXT	34082
データベースのデータをインポート...	IDC_IMPORT_DATABASE	34080
Microsoft Word 文書をインポート...	IDC_IMPORT_WORD	34083
データベース構造からXML スキーマを作成	IDC_CREATE_DB_SCHEMA	34054
XML スキーマを基にデータベースをインポート	IDC_IMPORT_DB_SCHEMA	34081
XML スキーマを基にデータベース構造を作成	IDC_CREATE_DB_BASED_ON_SCHEMA	34053
テキストファイルにエクスポート...	IDC_EXPORT_TEXTFILE	34064
データベースにエクスポート	IDC_EXPORT_DB	34003
XML インスタンスをJSON にから変換する...	IDC_JSON_CONVERT_TOFROM_XML	34135
XML スキーマを JSON スキーマにから変換する..	IDC_JSON_CONVERT_TOFROM_XSD	34350

25.4.6.11 「表示」メニュー

「表示」メニューには以下のコマンドが含まれます:

メニューアイテム	コマンド名	ID
テキストビュー	IDC_VIEW_TEXT	34180
拡張グリッドビュー	IDC_VIEW_GRID	34178
スキーマデザインビュー	IDC_VIEW_SCHEMA	34179
WSDL デザインビュー	IDC_VIEW_WSDL	34117
XBRL タクソノミビュー	IDC_VIEW_XBRL	34118
Authentic ビュー	IDC_VIEW_CONTENT	34177
ブラウザービュー	IDC_VIEW_BROWSER	34176
展開 +	IDC_SEL_EXPAND	34152
縮退 -	IDC_SEL_COLLAPSE	34151
全て展開	IDC_SEL_EXPAND_ALL	33463

メニューアイテム	コマンド名	ID
未選択を縮退	IDC_COLLAPSE_UNSELECTED	33428
最適な幅	IDC_OPTIMAL_WIDTHS	34099
ワードラップ	IDC_WORD_WRAP	34181
行/文字へ移動	IDC_GOTO_LINE	34073
ファイルへ移動	IDC_GOTO_FILE	33448
テキストビューの設定	IDC_TEXTVIEW_SETTINGS	34119

25.4.6.12 「ブラウザー」メニュー

「ブラウザー」メニューには以下のコマンドが含まれます:

メニューアイテム	コマンド名	ID
戻る	IDC_STEP_BACK	32958
進む	IDC_STEP_FORWARD	32957
停止	IDC_BROWSER_STOP	34047
最新の状態に更新する	IDC_BROWSER_REFRESH	34046
最大	IDC_BROWSER_FONT_LARGEST	34041
大	IDC_BROWSER_FONT_LARGE	34040
中	IDC_BROWSER_FONT_MEDIUM	34042
小	IDC_BROWSER_FONT_SMALL	34043
最小	IDC_BROWSER_FONT_SMALLEST	34044

25.4.6.13 「WSDL」メニュー

「W SDL」メニューには以下のコマンドが含まれます:

メニューアイテム	コマンド名	ID
メッセージの挿入	ID_WSDL_MESSAGES_ADDNEWMESSAGE	33715
メッセージの削除	ID_WSDL_MESSAGES_DELETESELECTED MESSAGE	33717
メッセージパートの追加（パラメーター）	ID_WSDL_MESSAGES_ADDMESSAGEPART	33714

(C) 2015-2021 Altova GmbH

メニューアイテム	コマンド名	ID
メッセージパートの削除（パラメーター）	ID_WSDL_MESSAGES_DELETEMESSAGEPART	33716
要求/応答	IDC_WSDL_OPERATION_APPENDREQUESTRESPONSE	33734
送信請求/応答	IDC_WSDL_OPERATION_APPENDSOLICITRESPONSE	33737
一方向	IDC_WSDL_OPERATION_APPENDONEWAY	33735
通知	IDC_WSDL_OPERATION_APPENDNOTIFICATION	33736
空のオペレーション	ID_WSDL_OPERATIONS_APPENDAOPERATIONTOTHISPORTTYPE	33722
オペレーションの削除	ID_WSDL_OPERATIONS_DELETEOPERATION	33724
input 要素の追加	ID_WSDL_OPERATIONS_ADDINPUTFUNCTION	33719
output 要素の追加	ID_WSDL_OPERATIONS_ADDOUTPUTFUNCTION	33721
fault 要素の追加	ID_WSDL_OPERATIONS_ADDFAULTFUNCTION	33718
input/output/fault 要素の削除	ID_WSDL_OPERATIONS_DELETEINPUTOUTPUTFUNCTION	33723
input/output/fault 要素に新しいメッセージを追加	ID_WSDL_OPERATIONS_ADDNEWMESSAGETOTHISELEMENT	33720
ポートタイプの挿入	ID_WSDL_PORTTYPE_INSERTAPORTTYPE	33727
ポートタイプの削除	ID_WSDL_PORTTYPE_DELETETHISPORTTYPE	33726
バインディングの挿入	ID_WSDL_BINDING_NEWBINDING	33713
バインディングの削除	ID_WSDL_BINDING_DELETEBINDING	33711
soap:body	ID_WSDL_BINDING_APPENDEXTENSIBILITY_SOAPBODY	33706
soap:header	ID_WSDL_BINDING_APPENDEXTENSIBILITY_SOAPHEADER	33708
soap:headerfault	ID_WSDL_BINDING_APPENDEXTENSIBILITY_SOAPHEADERFAULT	33709

メニューアイテム	コマンド名	ID
soap:fault	ID_WSDL_BINDING_APPENDEXTENSIBILITY_SOAPFAULT	33707
mime:content	ID_WSDL_BINDING_APPENDEXTENSIBILITY_MIMECONTENT	33702
mime:multipartrelated	ID_WSDL_BINDING_APPENDEXTENSIBILITY_MIMEMULTIPARTRELATED	33704
mime:part	ID_WSDL_BINDING_APPENDEXTENSIBILITY_MIMEPART	33705
mime:mimeXml	ID_WSDL_BINDING_APPENDEXTENSIBILITY_MIMEMIMEXML	33703
http:urlencoded	ID_WSDL_BINDING_APPENDEXTENSIBILITY_HTTPURLENCODED	33700
http:urlreplacement	ID_WSDL_BINDING_APPENDEXTENSIBILITY_HTTPURLREPLACEMENT	33701
拡張要素の削除	ID_WSDL_BINDING_DELETEEXTESIBILITY	33712
サービスの挿入	ID_WSDL_SERVICE_INSERTSERVICE	33731
サービスの削除	ID_WSDL_SERVICE_DELETETHISSERVICE	33729
ポートの挿入	ID_WSDL_SERVICE_INSERTNEWPORT	33730
ポートの削除	ID_WSDL_SERVICE_DELETETHISPORT	33728
新たなインターフェイスの追加	IDC_WSDL20_ADDINTERFACE	33794
インターフェイスの削除	IDC_WSDL20_DELETEINTERFACE	33795
fault 要素の追加	IDC_WSDL20_ADDINTERFACEFAULT	33796
fault 要素の削除	IDC_WSDL20_DELETEINTERFACEFAULT	33808
In-only	IDC_WSDL20_ADDINTERFACEOPERATION_INONLY	33797
Robust-in-only	IDC_WSDL20_ADDINTERFACEOPERATION_ROBUSTINONLY	33798
In-out	IDC_WSDL20_ADDINTERFACEOPERATION_INOUT	33801
In-opt-out	IDC_WSDL20_ADDINTERFACEOPERATION_INOPTOUT	33802
Out-in	IDC_WSDL20_ADDINTERFACEOPERATION_OUTIN	33803

(C) 2015-2021 Altova GmbH

メニューアイテム	コマンド名	ID
Out-opt-in	IDC_WSDL20_ADDINTERFACEOPERATION_OUTOPTIN	33804
Out-only	IDC_WSDL20_ADDINTERFACEOPERATION_OUTONLY	33800
Robust-out-only	IDC_WSDL20_ADDINTERFACEOPERATION_ROBUSTOUTONLY	33799
空のオペレーション	IDC_WSDL20_ADDINTERFACEOPERATION_EMPTY	33805
オペレーションの削除	IDC_WSDL20_DELETEINTERFACEOPERATION	33809
新たなバインディングの追加	IDC_WSDL20_ADDBINDING	33820
バインディングの削除	IDC_WSDL20_DELETEBINDING	33821
新規のfault バインディングの追加	IDC_WSDL20_ADDBINDINGFAULT	33822
fault バインディングの削除	IDC_WSDL20_DELETEBINDINGFAULT	33826
新たなオペレーションの追加	IDC_WSDL20_ADDBINDINGOPERATION	33823
オペレーションの削除	IDC_WSDL20_DELETEBINDINGOPERATION	33827
新たなサービスの追加	IDC_WSDL20_ADDSERVICE	33839
サービスの削除	IDC_WSDL20_DELETESERVICE	33840
新たなエンドポイントの追加	IDC_WSDL20_ADDENDPOINT	33841
エンドポイントの削除	IDC_WSDL20_DELETEENDPOINT	33842
新たなスキーマ	ID_WSDL_TYPES_NEWSCHEMA	33733
スキーマの埋め込み	ID_WSDL_TYPES_EMBEDSCHEMA	39456
スキーマの抽出	ID_WSDL_TYPES_EXTRACTSCHEMAS	39459
スキーマビューでのスキーマの編集	ID_WSDL_TYPES_EDITTHISSCHEMA	33732
図の保存...	IDC_WSDL_SAVE_DIAGRAM	39451
ドキュメントの生成	ID_WSDL_GENERATEDOCUMENTATION	39452
W SDL ドキュメントの再解析	IDC_WSDL_REPARSE	33774
W SDL 2.0 に変換する	IDC_WSDL_CONVERT_TO_WSDL20	39453
MapForce を使用してW SDL コードを生成する...	IDC_WSDL_GENERATE_CODE_MAPFORCE	34122

25.4.6.14 「SOAP」メニュー

「SOAP」メニューには以下のコマンドが含まれます:

メニューアイテム	コマンド名	ID
新規 SOAP リクエストを作成...	ID_SOAP_GENERATESOAPMESSAGE	34224
サーバーにリクエストを送信...	ID_SOAP_SENDREQUESTTOSERVER	34225
SOAP リクエスト設定...	ID_SOAP_SOAPREQUESTSETTINGS	34227
SOAP デバッガーセッション	ID_SOAP_SOAPDEBUGGER	34226
実行	ID_SOAPDEBUGGER_BUTTONPLAY	34221
ステップ実行	ID_SOAPDEBUGGER_SINGLESTEP	34222
次のリクエストでブレーク	ID_SOAPDEBUGGER_BREAKONNEXTREQUEST	34219
次のレスポンスでブレーク	ID_SOAPDEBUGGER_BREAKONNEXTRESPONSE	34220
プロキシサーバーの停止	ID_SOAPDEBUGGER_STOPSERVER	34223
SOAP デバッガーオプション	ID_SOAPDEBUGGEROPTIONS	34218

25.4.6.15 「XBRL」メニュー

「XBRL」メニューには以下のコマンドが含まれます:

メニューアイテム	コマンド名	ID
アークロール...	IDC_XMLSPYXBRLEDITOR_ARCROLES	34114
リンクロール...	IDC_XMLSPYXBRLEDITOR_LINKROLES	34115
名前スペースプレフィックス...	IDC_XMLSPYXBRLEDITOR_NAMESPACES	34116
ターゲット名前空間の設定...	IDC_XMLSPYXBRLEDITOR_SET_TARGETNAMESPACE	34039
パラメータ値...	IDC_ICXBRLEDITOR_PARAMETER_VALUES	38913
インポート/参照...	IDC_XMLSPYXBRLEDITOR_IMPORT_REFERENCE	34137
ID によりコンポーネントを検索...	IDC_ICXBRLEDITOR_FIND_COMPONENT_BY_ID	38893

(C) 2015-2021 Altova GmbH

メニューアイテム	コマンド名	ID
ドキュメントの生成	IDC_XMLSPYXBRLEDITOR_GENERATEDOCUMENTATION	34125
ビューの設定...	IDC_XMLSPYXBRLEDITOR_VIEWSETTINGS	34113
MapForce でDB、Excel、CSV からXBRL を生成...	IDC_XBRL_GENERATE_WITH_MAPFORCE	34045
StyleVision でHTML/PDF/Word としてXBRL を生成...	IDC_XBRL_PRESENT_WITH_STYLEVISION	34121
フォーミュラを実行する...	IDC_XBRL_EXECUTE_FORMULA	34305
フォーミュラをサーバーで実行する(ハイパフォーマンス)...	IDC_XBRL_EXECUTE_FORMULA_RAPTOR	34352
テーブルを生成する	IDC_XBRL_GENERATE_TABLE	34304
サーバーでテーブルを生成する(ハイパフォーマンス)...	IDC_XBRL_GENERATE_TABLE_RAPTOR	34353
インラインXBRL の変換	IDC_IXBRL_TRANSFORM	34354

25.4.6.16 「ツール」メニュー

「ツール」メニューには以下のコマンドが含まれています:

メニューアイテム	コマンド名	ID
スペル...	IDC_SPELL_CHECK	34154
スペルオプション...	IDC_SPELL_OPTIONS	34155
スクリプトエディター...	ID_SCRIPTFORMEDITOR_EDIT_PROJECT	39666
無し	ID_SCRIPTFORMEDITOR_EXECUTE_MACRO_MENU_UPPDATE	39600
開かれているファイルとの比較...	ID_XMLDIFF_CHOOSE_FILES	34235
ディレクトリの比較	ID_XMLDIFF_CHOOSE_DIRECTORIES	34234
比較のオプション	ID_XMLDIFF_SETTINGS	34236
	IDC_TOOLS_ENTRY	34292
グローバルリソース	IDC_GLOBALRESOURCES	37401
	IDC_GLOBALRESOURCES_SUBMENUENTRY1	37408
Raptor Servers の管理...	IDC_VALIDATE_RAPTOR_MANAGER	34311
無し	IDC_VALIDATE_RAPTOR_NOCFG	34326

メニューアイテム	コマンド名	ID
カスタマイズ...	IDC_APP_TOOLS_CUSTOMIZE	32959
オプション...	IDC_SETTINGS	34133
	ID_SCRIPTING_MACROITEMS	34249

25.4.6.17 「ヘルプ」メニュー

「ウィンドウ」メニューには以下のコマンドが含まれます:

メニューアイテム	コマンド名	ID
重ねて表示	ID_WINDOW_CASCADE	57650
上下に並べて表示	ID_WINDOW_TILE_HORZ	57651
左右に並べて表示	ID_WINDOW_TILE_VERT	57652
プロジェクトウィンドウ	IDC_PROJECT_WINDOW	34128
情報ウィンドウ	IDC_INFO_WINDOW	34085
入力ヘルパー	IDC_ENTRY_HELPERS	34062
出力ウィンドウ	IDC_OUTPUT_DIALOGBARS	34004
プロジェクトと入力ヘルパー	IDC_PROJECT_ENTRYHELPERS	34006
全てオン/オフ	IDC_ALL_BARS	34031

25.4.6.18 「ヘルプ」メニュー

「ヘルプ」メニューには以下のコマンドが含まれます:

メニューアイテム	コマンド名	ID
目次	IDC_HELP_CONTENTS	32966
インデックス	IDC_HELP_INDEX	32967
検索	IDC_HELP_SEARCH	32969
キーボードマップ	IDC_HELP_KEYMAPDLG	32968
ライセンスの登録	IDC_ACTIVATION	32970
注文フォーム	IDC_OPEN_ORDER_PAGE	32971
登録	IDC_REGISTRATION	32972

(C) 2015-2021 Altova GmbH

メニューアイテム	コマンド名	ID
最新情報のチェック	IDC_CHECK_FOR_UPDATES	32973
XMLSpy Product Comparison...	IDC_PRODUCT_COMPARISON	32955
サポートセンター	IDC_OPEN_SUPPORT_PAGE	32961
W eb のFAQ	IDC_SHOW_FAQ	32962
コンポーネントや無料ツールのダウンロード	IDC_OPEN_COMPONENTS_PAGE	32963
インターネット上のXMLSpy	IDC_OPEN_XML_SPY_HOME	32964
XMLSpy トレーニング	ID_HELP_XMLSPYTRAINING	32965
XMLSpy について	ID_APP_ABOUT	57664

25.4.7 オブジェクトレファレンス

オブジェクト:
XMLSpyCommand 1459
XMLSpyCommands 1461
XMLSpyControl 1462
XMLSpyControlDocument 1470
XMLSpyControlPlaceHolder 1476

標準のXMLSpy 機能へのアクセスを与えるには、**XMLSpy 自動化インターフェイス** のオブジェクトにもアクセスすることができます。詳細に関しては、次を参照してください: XMLSpyControl.Application 1463, XMLSpyControlDocument.Document 1471 と XMLSpyControlPlaceHolder.Project 1477 。

25.4.7.1 XMLSpyCommand

プロパティ:
ID 1460
Label 1460
Name 1461
IsSeparator 1460
ToolTip 1461
StatusText 1461
Accelerator 1460
SubCommands 1461

詳細:
コマンドオブジェクトは次のオプションであることができます: 実行可能なコマンド, コマンド コンテナー(例えば、メニュー、サブメニュー、または、ツールバー)、または、メニューセパレーター。現在の`Command` オブジェクト内に保管される情報の種類を決定するには、`ID`、`IsSeparator`、と`SubCommands` プロパティをクエリしてください。

コマンドオブジェクト	使用可能な場合...
実行可能なコマンド	• ID はゼロよりも大きい場合 • IsSeparator がfalse の場合 • SubCommands が空白の場合
コマンド コンテナー	• ID はゼロの場合 • IsSeparator がtrue の場合 • SubCommands がCommand オブジェクトを含む場合
セパレーター	• ID はゼロの場合 • IsSeparator がtrue の場合

25.4.7.1.1 Accelerator

プロパティ: Accelerator をstring として

詳細:
コマンドのために定義されたアクセラレータキーを返します。コマンドにアクセラレーターキーが割り当てられていない場合、このプロパティは、空の文字列を返します。 アクセラレータキーの文字列の表示は次のフォーマットを取ります:

```
[ALT+][CTRL+][SHIFT+]key
```

Windows プラットフォームSDK 関数 GetKeyNameText を使用して key が変換された箇所。

25.4.7.1.2 ID

プロパティ: ID をlong として

詳細:
このプロパティは、コマンドの一意の識別子を取得します。コマンドのID は Exec[1466] を使用して) コマンドを実行するために、または、(QueryStatus[1467] を使用して)ステータスをクエリするために必須です。コマンドが他のコマンド (例えば、トップレベルのメニュー)のためのコンテナー、または、セパレーターの場合、ID は0 です。

25.4.7.1.3 IsSeparator

プロパティ: IsSeparator をboolean として

詳細:
コマンドオブジェクトがメニューセパレーターの場合、の場合、プロパティは、 true を返します。それ以外の場合は、 false を返します。次も参照してください: コマンド[1459].

25.4.7.1.4 Label

プロパティ: Label をstring として

詳細:

このプロパティは、XMLSpy のグラフィカルなユーザーインターフェイス内に表示されるとおりのコマンドのテキストを取得します。コマンドがセパレーターの場合、「ラベル」は、空の文字列です。このプロパティは、関連した GUI テキストを持たない、一部のツールバーコマンドのために空の文字列を返します。

25.4.7.1.5 Name

プロパティ: `Name` を string として

詳細:
このプロパティは一意のコマンド名を取得します。使用できる箇所で、この値はコマンドのアイコンファイルを取得するために使用できます。使用することのできるアイコンファイルは、使用中の XMLSpy インストールのフォルダー **<ApplicationFolder>\Examples\ActiveX\Images** 内にあります。

25.4.7.1.6 StatusText

プロパティ: `Label` を string として

詳細:
コマンドが選択されると、ステータステキストは、XMLSpy のステータスバー内で表示されるテキストです。他のコマンドのセパレーターまたは、コンテナオではないコマンドオブジェクトに対してのみ適用することができます。それ以外の場合、プロパティは空の文字列です。

25.4.7.1.7 SubCommands

プロパティ: `SubCommands` を `Commands` [1461] として

詳細:
`SubCommands` プロパティは、現在のコマンドのサブコマンドである `Command` [1459] オブジェクトのコレクションを取得します。プロパティは、他のコマンド(メニュー、サブメニュー、または、ツールバー)のためのコンテナーであるコマンドにのみ適用されます。このようなコンテナーコマンドは、0 に設定された `ID` を持ち、および、`false` に設定された `IsSeparator` プロパティを持ちます。

25.4.7.1.8 ToolTip

プロパティ: `ToolTip` を string として

詳細:
このプロパティは、各コマンドのためのヒントを表示するテキストを取得します。コマンドがツールのヒントのテキストを持たない場合、プロパティは、空の文字列を返します。

25.4.7.2 XMLSpyCommands

プロパティ:
`Count` [1462]
`Item` [1462]

詳細:
XMLSpyControl のコマンドレベルと ID へのアクセスを取得するための `Command` [1459] オブジェクトのコレクションです。これらのコマンドは、`Exec` [1466] メソッドを使用して実行することができます。また、ステータスを `QueryStatus` [1467] を使用してクエリすることができます。

25.4.7.2.1 Count

プロパティ: Count を long として

詳細:
このコレクションのレベル上の Command [1459] オブジェクトの数。

25.4.7.2.2 Item

プロパティ: Item (n を long として) を Command [1459] として

詳細:
このコレクション内のインデックス n を持つコマンドを取得します。インデックスは 1 ベースです。

25.4.7.3 XMLSpyControl

プロパティ:
IntegrationLevel [1464]
Appearance [1463]
Application [1463]
BorderStyle [1463]
CommandsList [1464]
EnableUserPrompts [1464]
MainMenu [1465]
Toolbars [1465]

メソッド:
Open [1466]
Exec [1466]
QueryStatus [1467]

イベント:
OnUpdateCmdUI [1469]
OnOpenedOrFocused [1469]
OnCloseEditingWindow [1467]
OnFileChangedAlert [1468]
OnContextChanged [1468]
OnDocumentOpened [1468]
OnValidationWindowUpdated [1469]

このオブジェクトは完全な ActiveX コントロールで、XMLSpy ライブラリがアプリケーションレベルモードで使用されている場合、表示されます。

```
CLSID: a258bba2-3835-4c16-8590-72b44f52c471
ProgID: Altova.XMLSpyControl
```

25.4.7.3.1 プロパティ

以下のプロパティが定義されています:

IntegrationLevel 1464
EnableUserPrompts 1464
Appearance 1463
BorderStyle 1463

コマンドに関連したプロパティ:
CommandsList 1464
MainMenu 1465
Toolbars 1465

XMLSpyAPI へのアクセス:
Application 1463

25.4.7.3.1.1 Appearance

プロパティ: Appearance を short として

ディスパッチ ID : -520

詳細:
0 に対して等しくない値は、コントロールの周りでクライアントエッジを表示します。デフォルトの値は 0 です。

25.4.7.3.1.2 Application

プロパティ: Application を Application として

ディスパッチ ID : 1

詳細:
Application プロパティは、完全な XMLSpy 自動サーバーAPI の Application オブジェクトへのアクセスを与えます。プロパティは読み取り専用です

25.4.7.3.1.3 BorderStyle

プロパティ: BorderStyle を short として

ディスパッチ ID : -504

詳細:

1 の値は、薄いボーダーを持つコントロールを表示します。デフォルトの値は 0 です。

25.4.7.3.1.4 CommandsList

プロパティ: CommandList を Commands[1461] として (読み取り専用)

ディスパッチ ID : 1004

詳細:
このプロパティは、XMLSpyControl を持つ定義された全てのコマンドのフラットリストを返します。メニュー構造に従い、コマンドを整理するには、MainMenu[1465] を使用します。ツールバーコマンドを取得するには、Toolbars[1465] を使用します。

```
public void GetAllXmlSpyCommands()
{
    // Get all commands from the XMLSpy ActiveX control assigned to the current form
    XMLSpyControlLib.XMLSpyCommands commands = this.axXMLSpyControl1.CommandList;
    // Iterate through all commands
    for (int i = 0; i < commands.Count; i++)
    {
      // Get each command by index and output it to the console
      XMLSpyControlLib.XMLSpyCommand cmd = axXMLSpyControl1.CommandList[i];
      Console.WriteLine("{0} {1} {2}", cmd.ID, cmd.Name, cmd.Label.Replace("&", ""));
    }
}
```

C# サンプル

25.4.7.3.1.5 EnableUserPrompts

プロパティ: EnableUserPrompts を boolean として

ディスパッチ ID : 1006

詳細:
このプロパティを *false* に設定すると、コントロール内のユーザープロンプトを無効化します。デフォルトの値は *true* です。

25.4.7.3.1.6 IntegrationLevel

プロパティ: IntegrationLevel を ICActiveXIntegrationLevel[1479] として

ディスパッチ ID : 1000

詳細:
The IntegrationLevel プロパティ determines コントロールのオペレーションモード。次も参照してください: アプリケーションレベルの統合[1416] とドキュメントレベルの統合[1418] 。

メモ XMLSpyControl オブジェクトの作成後すぐに、このプロパティを設定することは重要です。

25.4.7.3.1.7 MainMenu

プロパティ: MainMenu を Command[1459] として(読み取り専用)

ディスパッチ ID : 1003

詳細:
このプロパティは、 XMLSpyControl メインメニュー内で使用することのできる構造の情報とコマンドを Command オブジェクトとして提供します。オブジェクトは、 XMLSpy の使用することのできる全てのサブメニューを含んでいます(例「ファイル」、「編集」、「表示」など)。サブメニューオブジェクトにアクセスするには、MainMenu プロパティの SubCommands プロパティを使用します。各サブメニューは Command オブジェクトでもあります。各サブメニューのために、対応する子コマンドセパレーターを取得するために、 SubCommands プロパティを使用して更に反復することができます(このテクニックが使用される可能性があります。例えば、アプリケーションメニューをプログラム的に作成するなど)。メニューコマンドの一部は、 SubCommands プロパティを持つ他のメニューコマンドのためのコンテナー(「親」)としての役割を果たします。全てのメニューコマンドの構造をプログラム的に取得するには、.再帰的な関数を作成する必要があります。

```
public void GetXmlSpyMenus()
{
    // Get the main menu from the XMLSpy ActiveX control assigned to the current form
    XMLSpyControlLib.XMLSpyCommand mainMenu = this.axXMLSpyControl1.MainMenu;

    // Loop through entries of the main menu (e.g. File, Edit, etc.)
    for (int i = 0; i < mainMenu.SubCommands.Count; i++)
    {
      XMLSpyControlLib.XMLSpyCommand menu = mainMenu.SubCommands[i];
      Console.WriteLine("{0} menu has {1} children items (including separators)",
menu.Label.Replace("&", ""), menu.SubCommands.Count);
    }
}
```

C# サンプル

25.4.7.3.1.8 Toolbars

プロパティ: Toolbars を Commands[1461] として (読み取り専用)

ディスパッチ ID : 1005

詳細:
このプロパティは、XMLSpyControl ツールバーの構造の情報 を Command オブジェクトとして与えます。Command オブジェクトには、XMLSpy の使用することのできる全てのツールバーが含まれています。ツールバーにアクセスするには、 Toolbars プロパティの SubCommands プロパティを使用します。各ツールバーは、 Command オブジェクトでもあります。各ツールバーのために、コマンドを使用するために SubCommands プロパティを更に反復します(例えば、アプリケーションのツールバープログラム的に作成するためなどにこのテクニックは使用される場合があります)。

```
public void GetXmlSpyToolbars()
{
    // Get the application toolbars from the StyleVision ActiveX control assigned to
the current form
```

```
    XMLSpyControlLib.XMLSpyCommands toolbars = this.axXMLSpyControl1.Toolbars;

    // Iterate through all toolbars
    for (int i = 0; i < toolbars.Count; i++)
    {
      XMLSpyControlLib.XMLSpyCommand toolbar = toolbars[i];
      Console.WriteLine();
      Console.WriteLine("The toolbar \"{0}\" has the following commands:",
toolbar.Label);

      // Iterate through all commands of this toolbar
      for (int j = 0; j < toolbar.SubCommands.Count; j++)
      {
         XMLSpyControlLib.XMLSpyCommand cmd = toolbar.SubCommands[j];
         // Output only command objects that are not separators
         if (!cmd.IsSeparator)
         {
            Console.WriteLine("{0}, {1}, {2}", cmd.ID, cmd.Name, cmd.Label.Replace("&",
 ""));
         }
      }
    }
}
```

C# サンプル

25.4.7.3.2 メソッド

以下のメソッドが定義されます:

Open 1466
Exec 1466
QueryStatus 1467

25.4.7.3.2.1 Exec

Method: `Exec` (`nCmdID` を long として) を boolean として

ディスパッチ ID : 6

詳細:
このメソッドは、ID `nCmdID` を持つ XMLSpy コマンドを呼び出し、コマンドが実行可能な場合、メソッドは `true` を返します。使用可能な全てのコマンドのリストを取得するには、`CommandsList` 1464 を使用してください。.コマンドの状態を取得するには、`QueryStatus` 1467 を使用してください。

25.4.7.3.2.2 Open

Method: `Open` (`strFilePath` を string として) を boolean として

ディスパッチ ID : 5

詳細:

メソッドの結果は、引数 strFilePath 内でパスされる拡張子により異なります。ファイル拡張子が .sps の場合、新規ドキュメントが開かれます。ファイル拡張子が.svp の場合、対応するプロジェクトが開かれます。異なるファイル拡張子がメソッドにパスされると、コントロールはファイルを新規のコンポーネントとしてアクティブなドキュメントにロードします。

ドキュメントレベルの統合モード内でコントロール使用する際、このメソッドを使用して、ドキュメントまたはプロジェクトをロードしないでください。代わりに、XMLSpyControlDocument.Open[1473] とXMLSpyControlPlaceHolder.OpenProject[1478] を使用してください。

25.4.7.3.2.3 QueryStatus

Method: QueryStatus (nCmdID を long として) を long として

ディスパッチ ID : 7

詳細:
QueryStatus は、nCmdID により指定されたコマンドのチェックされた/チェックが解除されたステータスを返します。このステータスはビットマスクとして返されます。

ビット	値	名前	意味
0	1	サポートされている	コマンドがサポートされている場合設定します。
1	2	有効化されている	コマンドが有効化されている場合設定します(実行可能)。
2	4	チェックされている	コマンドがチェックされている場合設定します。

これは、QueryStatus が0 コマンドを返すと、ID は、有効な XMLSpy コマンドとして認識されないことを意味します。QueryStatus が1 または5 の値を返すとコマンドは無効化されます。

25.4.7.3.3 Events

XMLSpyControl ActiveX コントロールは次の接続ポイントイベントを提供します:

OnUpdateCmdUI[1469]
OnOpenedOrFocused[1469]
OnCloseEditingWindow[1467]
OnFileChangedAlert[1468]
OnContextChanged[1468]

OnDocumentOpened[1468]
OnValidationWindowUpdated[1469]

25.4.7.3.3.1 OnCloseEditingWindow

イベント : OnCloseEditingWindow (i_strFilePath を String として) を boolean として

ディスパッチ ID : 1002

詳細:
XMLSpy が既に開かれているドキュメントを閉じる場合、このイベントはトリガーされます。このイベントへの応答として、クライアントは *i_strFilePath* に関連するエディターウィンドウを閉じます。このイベントが *true* を返すと、クライアントがドキュメントを閉じたことを指します。

特定の処理が必要とされておらず、XMLSpyControl がエディターを閉じようと試み、関連したドキュメントコントロールを破棄する場合、クライアント は、*false* を返すことができます。

25.4.7.3.3.2 OnContextChanged

イベント： `OnContextChanged` (`i_strContextName` を String として、`i_bActive` を bool として) を bool として

ディスパッチ ID：1004

詳細:
このイベントは XMLSpy が次のオペレーションコンテキストを有効化、または、無効化するとトリガーされます:

- XSLT プロファイル－"`XSLTProfiling`" がコンテキストメニューとして パスされます。
- XSLT / XQuery デバッグ－"`DebuggingXSLT`" がコンテキストメニューとして パスされます。

25.4.7.3.3.3 OnDocumentOpened

イベント： `OnDocumentOpened` (`objDocument` を Document として)

ディスパッチ ID：1

詳細:
このイベントはドキュメントが開かれる都度トリガーされます。引数 `objDocument` は、XMLSpy 自動化インターフェイス、からの `Document` オブジェクトであり、ドキュメントの詳細をクエリするために使用することができ、または、追加オペレーションを行います。ドキュメントレベルで統合される場合、イベント `XMLSpyControlDocument.OnDocumentOpened` 1475 を代わりに使用することが奨励されます。

25.4.7.3.3.4 OnFileChangedAlert

イベント： `OnFileChangedAlert` (`i_strFilePath` を String として) を bool として

ディスパッチ ID：1001

詳細:
XMLSpyControl を使用してファイルがロードされると、isこのイベントはトリガーされます。他のアプリケーションによりハードディスク上で変更されます。イベントを処理すると、クライアントは、true を返します。または、XMLSpy が通常の方法で処理すると、false を返します。例えば、ユーザーに再ロードを促すなど。

25.4.7.3.3.5 OnLicenseProblem

イベント： `OnLicenseProblem` (`i_strLicenseProblemText` を String として)

ディスパッチ ID：1005

詳細:
XMLSpyControl がこのコントロールのための有効なライセンスを使用できないことを検知すると、このイベントはトリガーされます。制限されているユーザーファイセンスの場合、コントロールが初期化されると発生する可能性があります。インテグレーターはこのイベントをしようして、この

コントロールの機能へのアクセスを無効化することができます。このイベントから戻ると、コントロールは機能へのアクセスをブロックします（例、コントロール内の空のウィンドウを表示し、リクエストに対してエラーを返すなど）。

25.4.7.3.3.6 OnOpenedOrFocused

イベント： `OnOpenedOrFocused (i_strFilePath` を String として、`i_bOpenWithThisControl` を bool として）

ディスパッチ ID： *1000*

詳細：
アプリケーションレベルで統合する場合、このイベントはクライアントにドキュメントが開かれたこと、またはXMLSpy によりアクティブに設定されたことを通知します。

ドキュメントレベルで統合する場合、このイベントは、クライアントにドキュメントウィンドウをファイル `i_strFilePath` で開くように命令します。ファイルが既に開かれている場合、対応するドキュメントウィンドウをアクティブなウィンドウにしてください。

`i_bOpenWithThisControl` が `true` の場合、内部アクセスが必要とされるため、ドキュメントは、XMLSpyControl により開かれる必要があります。それ以外の場合、ファイルは、異なるエディターで開くことができます。

25.4.7.3.3.7 OnToolWindowUpdated

イベント： `OnToolWindowUpdated(pToolWnd` を long として）

ディスパッチ ID： *1006*

詳細：
ツールウィンドウが更新されると、このイベントはトリガーされます。

25.4.7.3.3.8 OnUpdateCmdUI

イベント： `OnUpdateCmdUI ()`

ディスパッチ ID： *1003*

詳細：
頻繁に呼び出され、`XMLSpyControl.QueryStatus` [1467] を使用して XMLSpy コマンドの状況をチェックする良い機会をインテグレーターに与えます。このコールバック内で long オペレーションを実行しないでください。

25.4.7.3.3.9 OnValidationWindowUpdated

イベント： `OnValidationWindowUpdated ()`

ディスパッチ ID： *3*

詳細:
新しい情報と共に検証出力ウィンドウが更新されると、このイベントはトリガーされます。

25.4.7.4 XMLSpyControlDocument

プロパティ:
Appearance 1471
BorderStyle 1471
Document 1471
IsModified 1471
Path 1471
ReadOnly 1472

メソッド:
Exec 1472
New 1473
Open 1473
QueryStatus 1473
Reload 1473
Save 1474
SaveAs 1474

イベント:
OnDocumentOpened 1475
OnDocumentClosed 1475
OnModifiedFlagChanged 1476
OnContextChanged 1475
OnFileChangedAlert 1476
OnActivate 1474

XMLSpyControl がドキュメントレベルモード内で統合されている場合、各ドキュメントは、型 `XMLSpyControlDocument` のオブジェクト内で表示されています。`XMLSpyControlDocument` には、1度に1つのドキュメントのみが含まれていますが、場合によっては異なるファイルを表示するために使用される場合があります。

このオブジェクトは完全な ActiveX コントロールです。

```
CLSID: 52A552E6-2AB8-4e3e-B545-BE998233DDA0
ProgID: Altova.XMLSpyControlDocument
```

25.4.7.4.1 プロパティ

以下のプロパティが定義されています:

ReadOnly 1472
IsModified 1471
Path 1471
Appearance 1471
BorderStyle 1471

XMLSpyAPI へのアクセス:
Document 1471

25.4.7.4.1.1 Appearance

プロパティ: `Appearance` を short として

ディスパッチ ID : -520

詳細:
`0` に対して等しくない値は、ドキュメント コントロールの周りにクライアントエッジを表示します。デフォルトの値は `0` です。

25.4.7.4.1.2 BorderStyle

プロパティ: `BorderStyle` を short として

ディスパッチ ID : -504

詳細:
`1` の値は、薄いボーダー持つコントロールを表示します。デフォルトの値は `0` です。

25.4.7.4.1.3 Document

プロパティ: `Document` をドキュメントとして

ディスパッチ ID : 1

詳細:
`Document` プロパティは、XMLSpy 自動サーバーAPI の `Document` オブジェクト へのアクセスを与えます。このインターフェイスは、コントロール内にロードされるドキュメントと使用することのできる追加機能を与えます。プロパティは読み取り専用です

25.4.7.4.1.4 IsModified

プロパティ: `IsModified` を boolean として(読み取り専用)

ディスパッチ ID : 1006

詳細:
ドキュメントコンテンツが最後に開かれてから変更された場合、`IsModified` は *true* を返します。再ロードし、オペレーションを保存します。それ以外の場合は、`false` を返します。

25.4.7.4.1.5 Path

プロパティ: `Path` を string として

ディスパッチ ID : 1005

詳細:
コントロール内にロードされたドキュメントのフルパス名に設定、または、取得します。

25.4.7.4.1.6 ReadOnly

プロパティ: `ReadOnly` を boolean として

ディスパッチ ID : 1007

詳細:
このプロパティを使用すると、ドキュメントの読み取り専用モードをオンとオフに切り替えることができます。`ReadOnly` が `true` の場合、変更はできないことに注意してください。

25.4.7.4.2 メソッド

以下のメソッドが定義されています:

ドキュメントハンドリング:
New 1473
Open 1473
Reload 1473
Save 1474
SaveAs 1474

コマンドハンドリング::
Exec 1472
QueryStatus 1473

25.4.7.4.2.1 Exec

メソッド: `Exec (nCmdID` を long として) を boolean として

ディスパッチ ID : 8

詳細:
このメソッドは、ID `nCmdID` を持つ XMLSpy コマンドを呼び出し、コマンドが実行可能な場合、メソッドは、`true` を返します。現在アクティブなドキュメントがアプリケーションで使用できる場合のみ、このメソッドを呼び出すことができます。

メニュー構造に従いコマンドを整理するには、XMLSpyControl の `MainMenu` 1465 プロパティを使用します。ツールバーコマンドを取得するには、XMLSpyControl の `Toolbars` 1465 プロパティを使用します。

25.4.7.4.2.2 New

メソッド: `New()` を boolean として

ディスパッチ ID: 1000

詳細:
このメソッドはコントロール内で新規ドキュメントを初期化します。

25.4.7.4.2.3 Open

メソッド: `Open(strFileName` を string として) を boolean として

ディスパッチ ID: 1001

詳細:
`Open` は、ファイル `strFileName` を新規のドキュメントとして、コントロールにロードします。

25.4.7.4.2.4 QueryStatus

メソッド: `QueryStatus(nCmdID` を long として) を long として
ディスパッチ ID: 9

詳細:
`QueryStatus` は、`nCmdID` により指定されたコマンドのチェックされた/チェックが解除されたステータスを返します。このステータスはビットマスクとして返されます。

ビット	値	名前	意味
0	1	サポートされている	コマンドがサポートされている場合設定します。
1	2	有効化されている	コマンドが有効化されている場合設定します(実行可能)。
2	4	チェックされている	コマンドがチェックされている場合設定します。

これは、`QueryStatus` が `0` コマンドを返すと、ID は、有効な XMLSpy コマンドとして認識されないことを意味します。`QueryStatus` が `1` または `5` の値を返すとコマンドは無効化されます。アプリケーション内で現在アクティブなドキュメントが存在する場合、クライアントは、ドキュメントコントロールの `QueryStatus` メソッドを呼び出します。

25.4.7.4.2.5 Reload

メソッド: `Reload()` を boolean として

ディスパッチ ID: 1002

詳細:
`Reload` ファイルシステムからドキュメントコンテンツを更新する。

25.4.7.4.2.6 Save

メソッド: `Save()` を boolean として

ディスパッチ ID: 1003

詳細:
`Save` は、現在のドキュメントをロケーション `Path` [1471] に保存する。

25.4.7.4.2.7 SaveAs

メソッド: `SaveAs(strFileName` を string として) を boolean として

ディスパッチ ID: 1004

詳細:
`SaveAs` は `Path` [1471] を *strFileName* に設定し、ドキュメントをこの場所に保存します。

25.4.7.4.3 Events

XMLSpyControlDocumentActiveX コントロールは次の接続ポイントイベントを提供します:

`OnDocumentOpened` [1475]
`OnDocumentClosed` [1475]
`OnModifiedFlagChanged` [1476]
`OnContextChanged` [1475]
`OnFileChangedAlert` [1476]
`OnActivate` [1474]
`OnSetEditorTitle` [1476]

25.4.7.4.3.1 OnActivate

イベント: `OnActivate()`

ディスパッチ ID: 1005

詳細:
ドキュメントコントロールが有効化されると、フォーカスのある、ユーザー入力として使用することのできるこのイベントがトリガーされます。

(C) 2015-2021 Altova GmbH

25.4.7.4.3.2 OnContextChanged

イベント: `OnContextChanged` (`i_strContextName` を String として、`i_bActive` を bool として) を bool として

ディスパッチ ID: *1004*

詳細:
異なる XMLSpy ビュー内でドキュメントが表示されると、このイベントはトリガーされます。次の値がパスされます:

- グリッドビュー - `"View_0"` がコンテキストメニューとしてパスされます。
- テキストビュー - `"View_1"` がコンテキストメニューとしてパスされます。
- ブラウザービュー - `"View_2"` がコンテキストメニューとしてパスされます。
- スキーマビュー - `"View_3"` がコンテキストメニューとしてパスされます。
- Authentic ビュー - `"View_4"` がコンテキストメニューとしてパスされます。
- W SDL ビュー - `"View_5"` がコンテキストメニューとしてパスされます。

25.4.7.4.3.3 OnDocumentClosed

イベント: `OnDocumentClosed` (`objDocument` を Document として)

ディスパッチ ID: *1001*

詳細:
このイベントはドキュメントがこのコントロールにロードされると閉じられます。引数 `objDocument` は、XMLSpy 自動化インターフェイスからの `Document` オブジェクトです、で慎重に使用してください。

25.4.7.4.3.4 OnDocumentOpened

イベント: `OnDocumentOpened` (`objDocument` を Document として)

ディスパッチ ID: *1000*

詳細:
このイベントは、このコントロール内でドキュメントが開かれるとトリガーされます。引数 `objDocument` は、XMLSpy 自動化インターフェイスからの `Document` オブジェクトです。ドキュメントの詳細をクエリするために使用することができ、または、追加オペレーションを行います。

25.4.7.4.3.5 OnDocumentSaveAs

イベント: `OnContextDocumentSaveAs` (`i_strFileName` を String として)

ディスパッチ ID: *1007*

詳細:
このドキュメントが新しい名前で内部で保存される場合このイベントがトリガーされます。

25.4.7.4.3.6 OnFileChangedAlert

イベント: `OnFileChangedAlert` () を bool として

ディスパッチ ID: *1003*

詳細:
このドキュメントにロードされると、このイベントはトリガーされます。コントロールはハードディスク上で他のアプリケーションに変更されます。イベントを処理すると、クライアントは、true を返します。または、XMLSpy が通常の方法で処理すると、false を返します。例えば、ユーザーに再ロードを促すなど。

25.4.7.4.3.7 OnModifiedFlagChanged

イベント: `OnModifiedFlagChanged` (`i_bIsModified` を boolean として)

ディスパッチ ID: *1002*

詳細:
このイベントはドキュメントの状態を、変更された、および、変更されていない状態に切り替えると、トリガーされます。ドキュメントコンテンツが元のコンテンツと異なる場合、パラメーター *i_bIsModifed* is *true* それ以外の場合は、`false` を返します。

25.4.7.4.3.8 OnSetEditorTitle

イベント: OnSetEditorTitle ()

ディスパッチ ID: *1006*

詳細:
このイベントは、含まれているドキュメントが内部で名前を与えられると挙げられます。

25.4.7.5 XMLSpyControlPlaceHolder

すべてのプロジェクトプレースホルダーウィンドウに使用することのできるプロパティ:
PlaceholderWindowID 1477

プロジェクトプレースホルダーウィンドウのためのプロパティ:
Project 1477

プロジェクトプレースホルダーウィンドウのためのメソッド:
OpenProject 1478
CloseProject 1478

`XMLSpyControlPlaceHolder` コントロールは、概要、ライブラリまたはプロジェクトウィンドウなどの追加の XMLSpy ウィンドウを表示するために使用されます。他の ActiveX コントロールと同様に使用することができ、クライアントアプリケーション内に配置することができます。

`CLSID: 135DEEF4-6DF0-47c2-8F8C-F145F5F3F672`

(C) 2015-2021 Altova GmbH

ProgID: Altova.XMLSpyControlPlaceHolder

25.4.7.5.1 プロパティ

以下のプロパティが定義されています:

PlaceholderWindowID[1477]

XMLSpyAPI へのアクセス:
Project[1477]

25.4.7.5.1.1 Label

プロパティ: Label を String (読み取り専用)として

ディスパッチ ID: 1001

詳細:
このプロパティはプレースホルダのタイトルへのアクセスを与えます。プロパティは読み取り専用です

25.4.7.5.1.2 PlaceholderWindowID

プロパティ: PlaceholderWindowID を XMLSpyControlPlaceholderWindow[1479] として

ディスパッチ ID: 1

詳細:
このプロパティを使用すると、コントロールのクライアントエリア内で表示される XMLSpy ウィンドウが通知されます。PlaceholderWindowID は、XMLSpyControlPlaceholderWindow[1479] 列挙の有効な値に設定することができます。コントロールはすぐに状態を変更し、すぐに新規の XMLSpy ウィンドウを表示します。

25.4.7.5.1.3 Project

プロパティ: Project Project として(読み取り専用)

ディスパッチ ID: 2

詳細:
Project プロパティは、XMLSpy 自動サーバー API の Project オブジェクト へのアクセスを与えます。このインターフェイスは、コントロールへロードされるプロジェクトに使用される追加機能を与えます。プレースホルダウィンドウが XMLSpyXProjectWindow (=3) の値を持つ PlaceholderWindowID[1477] の場合のみ、プロパティは有効なプロジェクトインターフェイスを返します。プロパティは読み取り専用です

25.4.7.5.2 メソッド

以下のメソッドが定義されています:

OpenProject 1478
CloseProject 1478

25.4.7.5.2.1 OpenProject

メソッド: OpenProject (strFileName を string として) を boolean として

ディスパッチ ID : 3

詳細:
OpenProject は strFileName を新規のプロジェクトとしてコントロールにロードします。プレースホルダウィンドウが XMLSpyXProjectWindow (=3) と異なる PlaceholderWindowID 1477 を持つ場合、メソッドは失敗します。

25.4.7.5.2.2 CloseProject

メソッド: CloseProject ()

ディスパッチ ID : 4

詳細:
CloseProject はプロジェクトにロードされたコントロールを閉じます。プレースホルダウィンドウが XMLSpyXProjectWindow (=3) と異なる PlaceholderWindowID 1477 を持つ場合、メソッドは失敗します。

25.4.7.5.3 イベント

XMLSpyControlPlaceholder ActiveX コントロールは次の接続ポイントイベントを提供します:

OnModifiedFlagChanged 1478

25.4.7.5.3.1 OnModifiedFlagChanged

イベント: OnModifiedFlagChanged (i_bIsModified を boolean として)

ディスパッチ ID : 1

詳細:
このイベントは XMLSpyXProjectWindow (=3) の PlaceholderWindowID 1477 を持つプレースホルダコントロールのためのみにトリガーされます。プロジェクトのコンテンツが変更された、または、変更されていない状態の間で変更があった場合、イベントは実行されます。プロジェクトコンテンツが元のコンテンツと異なる場合、パラメーター *i_bIsModifed* は *true* を返し、それ以外の場合は、false を返します。

(C) 2015-2021 Altova GmbH

25.4.7.5.3.2 OnSetLabel

イベント: `OnSetLabel(i_strNewLabel` を string として)

ディスパッチ ID: 1000

詳細:
プレースホルダウィンドウのタイトルが変更されると、挙げられます。

25.4.7.6 列挙

以下の列挙が定義されています:

ICActiveXIntegrationLevel 1479
XMLSpyControlPlaceholderWindow 1479

25.4.7.6.1 ICActiveXIntegrationLevel

XMLSpyControl の IntegrationLevel 1464 プロパティのための可能な値。

```
ICActiveXIntegrationOnApplicationLevel  = 0
ICActiveXIntegrationOnDocumentLevel     = 1
```

25.4.7.6.2 XMLSpyControlPlaceholderWindow

この列挙にはサポートされる追加 XMLSpy ウィンドウのリストが含まれています。

```
XMLSpyControlNoToolWnd                          = -1
XMLSpyControlEntryHelperTopToolWnd              = 0
XMLSpyControlEntryHelperMiddleToolWnd           = 1
XMLSpyControlEntryHelperBottomToolWnd           = 2
XMLSpyControlValidatorOutputToolWnd             = 3
XMLSpyControlProjectWindowToolWnd               = 4
XMLSpyControlXSLTDebuggerContextToolWnd         = 5
XMLSpyControlXSLTDebuggerCallstackToolWnd       = 6
XMLSpyControlXSLTDebuggerVariableToolWnd        = 7
XMLSpyControlXSLTDebuggerWatchToolWnd           = 8
XMLSpyControlXSLTDebuggerTemplateToolWnd        = 9
XMLSpyControlXSLTDebuggerInfoToolWnd            = 10
XMLSpyControlXSLTDebuggerMessageToolWnd         = 11
XMLSpyControlXSLTDebuggerTraceToolWnd           = 12
XMLSpyControlSOAPDebuggerToolWnd                = 13
XMLSpyControlXPathProfilerListToolWnd           = 14
XMLSpyControlXPathProfilerTreeToolWnd           = 15
XMLSpyControlXPathDialogToolWnd                 = 16
XMLSpyControlDBQueryManagerToolWnd              = 17
```

```
XMLSpyControlInfoToolWnd                    = 18
XMLSpyControlXSLOutlineToolWnd              = 19
XMLSpyControlSchemaFindToolWnd              = 20
XMLSpyControlXBRLFindToolWnd                = 21
XMLSpyControlChartsToolWnd                  = 22
```


(C) 2015-2021 Altova GmbH

26 付録

以下の付録には、XMLSpy に関する技術的な情報や、ライセンスに関する重要な情報が収められています。各付録には以下のようにサブセクションが収められています:

エンジン情報 1482

- XSLT とXQuery エンジン情報 1482
- XSLT とXQuery 拡張関数 1488

データ型、DB とXML スキーマ間の変換

- DB からXML スキーマへの変換 1578
- XML スキーマからDB への変換 1584

技術情報 1591

- OS ならびにメモリの必要条件 1591
- Altova XML パーサー 1591
- Altova XSLT ならびにXQuery エンジン 1591
- Unicode のサポート 1591
- インターネットへの接続 1592

ライセンス情報 1593

- 電子的なソフトウェアの配布 1593
- ソフトウェアのアクティベーションとライセンスメータリング 1593
- エンドユーザー使用許諾契約書 1594

26.1 XSLT および XQuery エンジンに関する情報

XMLSpy のXSLT およびXQuery エンジンは、W 3C 仕様に従っています、ですから、XMLSpy の以前のバージョン内のAltova エンジンより先厳密です。この結果、以前のエンジンで無視されていた小さなエラーが、XMLSpy によりエラーとして挙げられます。

例えば

- パス演算子の結果がノードと非-ノードを両方含む場合、型エラー(`err:XPTY0018`) です。
- パス式 `E1/E2` 内の`E1` がノードのシーケンスを評価しない場合、型エラー(`err:XPTY0019`) です。

この種類のエラーが発生した場合、XSLT/XQuery ドキュメントまたはインスタンスドキュメントを必要に応じて修正してください。

このセクションは、エンジンの実装固有の機能を仕様別に整理して説明します。

- XSLT 1.0 1482
- XSLT 2.0 1482
- XSLT 3.0 1484
- XQuery 1.0 1484
- XQuery 3.1 1487

26.1.1 XSLT 1.0

XMLSpy のXSLT 1.0 エンジンは、W orld W ide W eb Consortium(ワールド・ワイド・ウェブ・コンソーシアム) (W 3C) の1999 年 11 月 16 日版のXSLT 1.0 勧告 および1999 年 11 月 16 日版のXPath 1.0 勧告 に準拠します。実装に関しての以下の情報に注意してください。

実装についての注意点

`xsl:output` の`method` 属性がHTMLに設定された場合、または、がHTML 出力 デフォルトで選択されている場合、XML またはXSLT ファイル内の特殊文字はHTML ドキュメントにHTML 文字参照として出力内に挿入されます。例えば、文字 U+00A0 (ブレーク無しのスペースのための16 進数レファレンス) がHTML コード内に文字の参照(` ` or ` `) 、または、エンティティ参照 ` ` として挿入されます

26.1.2 XSLT 2.0

このセクション:

- エンジン適合性 1483
- 下位互換性 1483
- 名前空間 1483
- スキーマ認識 1483
- 実装固有の振る舞い 1484

適合性

XMLSpy のXSLT 2.0 エンジンは、W orld W ide W eb Consortium（ワールド・ワイド・ウェブ・コンソーシアム）（W 3C）の2007 年 1 月 23 日版のXSLT 2.0 勧告 および2010 年 12 月 14 日版のXPath 2.0 勧告に準拠します。

下位互換性

XSLT 2.0 エンジン は下位互換性を有します。XSLT 2.0 エンジンの 下位互換性が有効になるのは、 XSLT 1.0 スタイルシートを処理するためにXSLT 2.0 エンジン が使用される際です。XSLT 1.0 エンジン と下位互換性を持つXSLT 2.0 エンジンにより作成される出力に違いがあるかもしれないことに注意してください。

名前空間

XSLT 2.0 スタイルシートは、 XSLT 2.0. プレフィックス内で使用することのできる型コンストラクタ および関数を使用するため、以下の名前空間を宣言する必要があります。 下のリストは通常使用されるリスト です。希望する場合は、代替プレフィックスを使用することもできます。

名前空間	プレフィックス	名前空間 URI
XML スキーマ型	xs:	http://www.w3.org/2001/XMLSchema
XPath 2.0 関数	fn:	http://www.w3.org/2005/xpath-functions

通常これらの名前空間は、以下のリスト で表示されるようにxsl：スタイルシートまたは、 xsl:transform 要素で宣言されます:

```
<xsl：スタイルシート version="2.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:fn="http://www.w3.org/2005/xpath-functions"
    …
</xsl：スタイルシート>
```

次の点に注意してください:

- XSLT 2.0 エンジンは、（上のテーブルでリストされている）XPath 2.0 およびXQuery 1.0 関数 名前空間 をデフォルト の関数名前空間として使用します。 XPath 2.0 およびXSLT 2.0 関数をプレフィックス無しでスタイルシート内で使用することができます。XPath 2.0 関数 名前空間 をスタイルシート内でプレフィックスと共に宣言すると、 割り当てられた宣言内でプレフィックスを追加して使用することができます。
- XML スキーマ 名前空間から型コンストラクタと型を使用する場合、名前空間 宣言内で使用された、プレフィックスを使用して型コンストラクタを呼び出さなければなりません（例えば、 xs:date）。
- XPath 2.0 関数の一部は、XML スキーマ データ型と同じ名前を保有します。 例えば、 XPath 関数 fn:string および fn:boolean のためには、同じロケーション名: xs:string および xs:boolean を持つ、XML スキーマ データ型 が存在します。ですから、 XPath 式 string('Hello') を使用する場合、 式は、 xs:string('Hello') ではなくて fn:string('Hello') として検証します。

スキーマ認識

XSLT 2.0 エンジンは、スキーマを認識します。ですから、ユーザー定義 スキーマ型 および xsl:validate 命令を使用することができます。

実装固有の振る舞い

以下は、XSLT 2.0 エンジンが、特定のXSLT 2.0 関数の振る舞いの実装-特定のアスペクトをどのように扱うかの説明です。

xsl:result-document
追加してサポートされるエンコードは以下の通りです（Altova-固有）: `x-base16tobinary` および `x-base64tobinary`.

function-available
インスコープ関数の使用をテストする関数（XSLT、XPath、および拡張関数）。

unparsed-text
`href` 属性は、以下を受け入れます（i）ベース-uri フォルダー内のファイルの相対パス、および（ii） 相対パスを持つまたは持たない `file://` プロトコール。追加してサポートされるエンコードは以下の通りです（Altova-固有）: `x-binarytobase16` および `x-binarytobase64`.

unparsed-text-available
`href` 属性は、以下を受け入れます（i）ベース-uri フォルダー内のファイルの絶対パス、および（ii）絶対パスを持つまたは持たない `file://` プロトコール。追加してサポートされるエンコードは以下の通りです（Altova-固有）: `x-binarytobase16` および `x-binarytobase64`.

メモ RaptorXML の先行製品であるAltovaXML で実装されていた以下のエンコード値は使用しないでください: `base16tobinary`, `base64tobinary`, `binarytobase16` and `binarytobase64`.

26.1.3 XSLT 3.0

XMLSpy のXSLT 3.0 エンジンは、World Wide Web Consortium（ワールド・ワイド・ウェブ・コンソーシアム）（W3C）の2017 年 6 月 8 日版 XSLT 3.0 勧告候補 および2017 年 3 月 21 日版 XPath 3.1 勧告候補 に準拠します。

XSLT 3.0 エンジンは、XSLT 2.0 エンジンと同様の実装固有の機能を搭載しています。更に、以下のXSLT 3.0 機能へのサポートを含みます。更に次の一連の新規のXSLT 3.0 機能をサポートします: XPath/XQuery 3.1 関数とオペレーターとXPath 3.1 仕様。

メモ 任意のストリーミングの機能は現在サポートされていません。streamable 属性の値に関係なく、ドキュメント全体がメモリにロードされ、使用することのできるメモリが十分な場合は処理されます。64-ビットアプリでは、これは問題になりません。thisメモリが問題の場合は、システムに解決策を追加する必要があります。

26.1.4 XQuery 1.0

このセクション:

- エンジン適合性 1485
- スキーマ認識 1485
- エンコード 1485
- 名前空間 1483
- XML ソースと検証 1486
- 静的および動的な型のチェック 1486
- ライブラリモジュール 1486
- 外部関数 1486

- 照合順序 1486
- 数値データの精度 1486
- XQuery 命令サポート 1487

適合性

XMLSpy のXQuery 1.0 エンジンは、W orld W ide W eb Consortium（ワールド・ワイド・ウェブ・コンソーシアム）（W 3C）の2010 年 12 月 14 日版のXQuery 1.0 勧告に準拠します。XQuery 標準は、多数の機能の実装についての裁量を提供します。下には、XQuery 1.0 エンジンがどのようにこれらの機能を実装するかについて説明するリストが下に挙げられています。

スキーマ認識

XQuery 1.0 エンジン はスキーマを認識します。

エンコード

UTF-8 およびUTF-16 文字のエンコードは、サポートされています。

名前空間

以下の名前空間 URI と関連するバインドは定義済みです。

名前空間	プレフィックス	名前空間 URI
XML スキーマ型	xs:	http://www.w3.org/2001/XMLSchema
スキーマインスタンス	xsi:	http://www.w3.org/2001/XMLSchema-instance
内蔵の関数	fn:	http://www.w3.org/2005/xpath-functions
Local 関数	local:	http://www.w3.org/2005/xquery-local-functions

次の点に注意してください：

- XQuery 1.0 エンジンは、上にリストされたプレフィックスを名前空間に対応するバウンドとして認識します。
- Since the 上にリストされた内蔵の関数 名前空間は、XQuery内のデフォルトの関数です。内蔵の関数 が呼び出される際、名前空間、fn: プレフィックス を使用する必要はありません。（例えば、string("Hello") がfn:string 関数を呼び出す場合。）しかし、プレフィックス fn: は、クエリプロローグ内で名前空間を宣言することなく内蔵の関数を呼び出す時に使用することができます。（サンプル: fn:string("Hello")）.
- クエリプロローグ内でdefault function 名前空間 式を宣言することにより、デフォルトの関数 名前空間をすることにより変更することができます。
- XML スキーマ 名前空間空の型を使用する場合、プレフィックス xs: は、名前空間を明確に宣言することなく、また、これらのプレフィックスをクエリプロローグ内でバインドすることなく使用することができます。（サンプル: xs:date および xs:yearMonthDuration。）XML スキーマ 名前空間のために、他のプレフィックスを使用する場合は、クエリプロローグ内で明確に宣言されている必要があります。（サンプル: declare 名前空間 alt = "http://www.w3.org/2001/XMLSchema"; alt:date("2004-10-04").）
- untypedAtomic, dayTimeDuration, およびyearMonthDuration データ型 が23 January 2007 のCR 共に、XPath データ型 名前空間からXML スキーマ 名前空間へ移動されていることに注意してください。ですから以下となります: xs:yearMonthDuration。

関数のための名前空間、型コンストラクタ、ノード テスト、が間違って割り当てられている場合、エラーが発生します。しかし、一部の関数はスキーマデータ型と同じ名前を持つことに注意してください。 例: fn:string および fn:boolean。(xs:string および xs:boolean は宣言されています) 名前空間 プレフィックス、関数または 型コンストラクタが使用されるか決定します。

XML ソースドキュメントと検証

XQuery 1.0 エンジンを使用して、実行される XQuery ドキュメント内の XML ドキュメントは、整形式である必要があります。しかし、XML スキーマに従い有効である必要はありません。XML ファイルが外部スキーマと関連付けられ、また有効な場合、ポスト スキーマ検証情報が XML データのために生成され、クエリ検証のために使用されます。

静的および動的な型のチェック

静的分析フェーズは、外部レファレンスの存在（例、モジュールのため）、呼び出された関数と変数が定義済みであるか、など構文などのクエリのアスペクトをチェックします。静的分析フェーズでエラーが検知されると、実行は停止されます。

クエリが実際に作動中にランタイム中に動的な型チェックは実行されます。型がオペレーションの必要条件と整合性を持たない場合、エラーが報告されます。例えば、式 `xs:string("1") + 1` は、エラーを返します。型 `xs:string` のオペランドを足し算のオペレーションが実行できないためです。

ライブラリモジュール

ライブラリモジュールは、再利用のため関数と変数を保管します。XQuery 1.0 エンジンは単一の外部 XQuery ファイルに保管されているモジュールをサポートします。このようなモジュール ファイルはプロローグにターゲット名前空間に関連するモジュール宣言を含む必要があります。 以下はモジュールサンプルです:

```
module namespace libns="urn:module-library";
declare variable $libns:company := "Altova";
declare function libns:webaddress() { "http://www.altova.com" };
```

すべての関数および変数は、モジュールに関連した名前空間に属するモジュール内で宣言されています。モジュールはクエリプロローグ内の `import module` ステートメントを使用して XQuery ファイルにインポートする際に使用されます。`import module` ステートメントは、ライブラリモジュールファイル内で直接宣言された、関数と変数のみをインポートします。例:

```
import module namespace modlib = "urn:module-library" at "modulefilename.xq";
if        ($modlib:company = "Altova")
then       modlib:webaddress()
else       error("No match found.")
```

外部関数

外部関数 は、サポートされていません。例えば、`external` キーワードを仕様する式など。以下参照:

```
declare function hoo($param as xs:integer) as xs:string external;
```

照合順序

デフォルトの照合順序は、Unicode コードポイントをベースにした文字列を比較する Unicode-コードポイント照合順序です。その他にサポートされる照合順序は、ICU 照合順序 はここ 1488 にリストされるとおりです。特定の照合順序を使用する場合、サポートされる照合順序のリスト 1488 内に与えられているとおり URI を提供します。`fn:max` と `fn:min` 関数を含む文字列の比較は、指定された照合順序により行われます。 照合順序オプションが指定されていない場合、デフォルトの Unicode-コードポイント照合順序 が使用されます。

数値データの精度

- `xs:integer` データ型には任意の精度があり、表記できる桁数に制限はありません。
- `xs:decimal` データ型には小数点の後に 20 桁の制限があります。
- `xs:float` と `xs:double` データ型には 15 桁の精度の制限があります。

(C) 2015-2021 Altova GmbH

XQuery 命令サポート

`Pragma` 命令 は、サポートされていません。発生した場合、無視されフォールバックの式が検証されます。

26.1.5 XQuery 3.1

XMLSpy のXQuery 3.1 エンジンは、W orld W ide W eb Consortium（ワールド・ワイド・ウェブ・コンソーシアム）（W 3C）の2017年 3 月 21 日版のXQuery 3.1 候補勧告に準拠し、またXPath およびXQuery 関数 3.1. へのサポートを含みます。XQuery 3.1 仕様は、3.0 仕様のスーパーセットです。XQuery 3.1 エンジンは、ですかXQuery 3.0 機能をサポートします。

実装固有の特性はXQuery 1.0[1484]同様です。

26.2 XSLT と XPath/XQuery 関数

このセクションでは、XPath および/または XQuery 式で使用することのできる、Altova 拡張関数と他の拡張関数をリストします。Altova 拡張関数は Altova の XSLT および XQuery エンジンで使用することができ、W3C 標準で定義された関数ライブラリで使用することのできる機能に追加して機能を提供します。

一般的な情報

以下の一般的な情報に注意してください：

- W3C 仕様により定義されているコア関数ライブラリの関数は、関数の呼び出しにプレフィックスは必要ありません。これは、XSLT および XQuery エンジンが、XPath/XQuery 関数仕様で指定されている `http://www.w3.org/2005/xpath-functions` プレフィックス無しの関数をデフォルト関数の名前空間に属するものとして読み込むためです。
- 関数において、各アイテムが引数となるようなシーケンスが期待されており、2つ以上のアイテムがシーケンスにより呼び出された場合、エラーが返されます。
- 全ての比較は Unicode コードポイントコレクションを使用することで行われます。
- QName の結果は `[prefix:]localname` という形式でシリアル化されます。

xs:decimal の精度

精度とは、数値内にある桁数のことで、仕様では少なくとも18桁が求められます。`xs:decimal` 型に結果が収められる除算の場合、端数処理を行うことなく精度は小数点以下の19桁になります。

黙示的なタイムゾーン

2つの `date`、`time`、または `dateTime` 値を比較する場合、比較する値のタイムゾーンを明らかにする必要があります。値の中にタイムゾーンが明示的に与えられていない場合、黙示的なタイムゾーンが使用されます。黙示的なタイムゾーンはシステムクロックから取得され `implicit-timezone()` 関数によりその値をチェックすることができます。

照合順序

デフォルトの照合順序は、Unicode コードポイントをベースに文字列を比較する Unicode コードポイント照合順序です。エンジンは Unicode 照合アルゴリズムを使用しています。他のサポートされる照合順序は下にリストされる ICU 照合順序 です。使用するには、サポートされる照合順序のリストの URI を提供してください（*下のテーブル*）。`max` と `min` 関数を含む、文字列の比較は、指定された照合順序に沿って行われます。照合順序オプションが指定されていない場合、デフォルトの Unicode コードポイント照合順序が使用されます。

言語	URI
`da`: デンマーク語	`da_DK`
`de`: ドイツ語	`de_AT, de_BE, de_CH, de_DE, de_LI, de_LU`
`en`: 英語	`en_AS, en_AU, en_BB, en_BE, en_BM, en_BW, en_BZ, en_CA, en_GB, en_GU, en_HK, en_IE, en_IN, en_JM, en_MH, en_MP, en_MT, en_MU, en_NA, en_NZ, en_PH, en_PK, en_SG, en_TT, en_UM, en_US, en_VI, en_ZA, en_ZW`
`es`: スペイン語	`es_419, es_AR, es_BO, es_CL, es_CO, es_CR, es_DO, es_EC, es_ES, es_GQ, es_GT, es_HN, es_MX, es_NI, es_PA, es_PE, es_PR, es_PY, es_SV, es_US, es_UY, es_VE`
`fr`: フランス語	`fr_BE, fr_BF, fr_BI, fr_BJ, fr_BL, fr_CA, fr_CD, fr_CF, fr_CG, fr_CH, fr_CI, fr_CM, fr_DJ, fr_FR, fr_GA, fr_GN, fr_GP, fr_GQ, fr_KM, fr_LU, fr_MC, fr_MF, fr_MG, fr_ML, fr_MQ, fr_NE, fr_RE, fr_RW, fr_SN, fr_TD, fr_TG`

(C) 2015-2021 Altova GmbH

it: イタリア語	it_CH, it_IT
ja: 日本語	ja_JP
nb: ノルウェー語 (ブークモール)	nb_NO
nl: オランダ語	nl_AW, nl_BE, nl_NL
nn: ノルウェー語 (ニーノシュク)	nn_NO
pt: ポルトガル語	pt_AO, pt_BR, pt_GW, pt_MZ, pt_PT, pt_ST
ru: ロシア語	ru_MD, ru_RU, ru_UA
sv: スウェーデン語	sv_FI, sv_SE

名前空間軸
名前空間軸はXPath 2.0 にて廃止されましたが、名前空間軸の使用はサポートされています。XPath 2.0 メカニズムにより名前空間情報へアクセスするには、`in-scope-prefixes()`、`namespace-uri()`、`namespace-uri-for-prefix(` 関数を使用してください。

26.2.1 Altova 拡張関数

Altova 拡張関数はXPath/XQuery 式で使用することができ、XPath、XQuery、およびXSLT 関数の標準ライブラリで使用可能な機能に更なる機能性を与えます。Altova 拡張関数はAltova **拡張関数名前空間、`http://www.altova.com/xslt-extensions`** に収められており、**`altova:`** プレフィックスが、このセクションでは使用されます。製品の今後のバージョンが拡張機能への継続的サポート、または個別の関数の振る舞いは変更する可能性があることに注意してください。Altova 拡張機能へのサポートに関しては、今後のリリースのドキュメントを参照してください。

W 3C のXPath/XQuery 関数仕様で定義された関数は、以下で使用することができます: (i) XSLT 子アンテキスト内のXPath 式と、(ii) XQuery 文書内のXQuery 式。このドキュメントでは、前者(XSLT 内のXPath) のコンテキストで使用することのできる関数を、**XP** シンボルと共に表示し、と称します。後者(XQuery) で使用することのできる関数はの**XQ** シンボルと共に表示され、XQuery 関数と共に作業することができます。W 3C のXSLT 仕様は、XPath/XQuery 関数の仕様ではなく、XSLT 文書内のXPath 式でも使用することのできる関数を定義します。これらの関数は、**XSLT** シンボルと共に表示され、XSLT 関数と称されます。関数を使用することのできるXPath/XQuery およびXSLT のバージョンは、関数の詳細に記載されています (*下のシンボルを参照してください*)。
XPath/XQuery およびXSLT 関数ライブラリからの関数は、プレフィックス無しでリストされています。Altova 拡張関数などの、他のライブラリからの関数はプレフィックスと共にリストされています。

XPath 関数 (XSLT 内のXPath 式で使用):	**XP1** **XP2** **XP3.1**
XSLT 関数 (XSLT 内のXPath 式で使用):	**XSLT1** **XSLT2** **XSLT3**
XQuery 関数 (XQuery 内のXQuery 式で使用):	**XQ1** **XQ3.1**

XSLT 関数 1490

XSLT 関数はXSLT 2.0 の`current-group()` や`key()` 関数と同様に、XSLT コンテキストにて使用することができます。(例えば、XQuery コンテキストなどの) 非-XSLT コンテキストでは使用することができません。XBRL に対するXSLT 関数は、XBRL をサポートするエディションのAltova 製品でのみ使用することができます。

XPath/XQuery 関数

XPath/XQuery 関数は、 XSLT コンテキスト、 XQuery 関数のXPath 式で使用することができます:

- 日付/時刻 1492
- 位置情報 1507
- イメージに関連した 1517
- 数値 1526
- シーケンス 1546
- 文字列 1553
- その他 1559

26.2.1.1 XSLT 関数

XSLT 拡張関数 はXSLT コンテキスト内のXPath 式にて使用することができます。(例えば、XQuery コンテキストなどの) 非-XSLT コンテキストでは使用することができません。

関数の名前指定と言語の適用性に関するメモ

Altova 拡張関数はXPath/XQuery 式で使用することができ、XPath、XQuery、およびXSLT 関数の標準ライブラリで使用可能な機能に更なる機能性を与えます。Altova 拡張関数はAltova **拡張関数名前空間**、**`http://www.altova.com/xslt-extensions`** に収められており、**`altova:`** プレフィックスが、このセクションでは使用されます。製品の今後のバージョンが拡張機能への継続的サポート、または個別の関数の振る舞いは変更する可能性があることに注意してください。Altova 拡張機能へのサポートに関しては、今後のリリースのドキュメントを参照してください。

XPath 関数 (XSLT 内のXPath 式で使用):	XP1 XP2 XP3.1
XSLT 関数 (XSLT 内のXPath 式で使用):	XSLT1 XSLT2 XSLT3
XQuery 関数 (XQuery 内のXQuery 式で使用):	XQ1 XQ3.1

標準関数

distinct-nodes [altova:]

`altova:distinct-nodes(`*`node()*`***`)`** を **`node()*`** とする XSLT1 XSLT2 XSLT3

入力として1つ以上のノードを必要とし、同じセットから重複した値を持つノードを除いたノードを返します。XPath/XQuery 関数 `fn:deep-equal` を使用して比較を行うことができます。

サンプル

- **`altova:altova:distinct-nodes`**`(country)` は重複した値を持つものを除く、全ての子 `country` ノード返します。

evaluate [altova:]

`altova:evaluate(XPathExpression` *`as xs:string`*`[, ValueOf$p1, ... ValueOf$pN])` XSLT1

XSLT2 **XSLT3**

XPath 式を必要とし、必須引数として文字列をパスします。評価された式の出力を返します。例えば **altova:evaluate('//Name[1]')** は、ドキュメント内の最初の Name 要素のコンテンツを返します。式 //Name[1] は、一重引用符を使用することにより、文字列としてパスされます。

altova:evaluate 関数は、オプションとして追加の引数を持つことができます。これらの引数は、p1, p2, p3... pN の名前を持つスコープ内の変数の値です。使用に関して以下の点に注意してください：(i) 変数は、x が整数である箇所のフォーム pX の名前と共に定義される必要があります。(ii) altova:evaluate 関数の引数は、(*上の署名参照*) 2 番目の引数からは、数値順の変数のシーケンスに対応した引数のシーケンス変数の値を与えます: p1 to pN: 第 2 引数は変数 p1 の値で、第 3 引数は、変数 p2 の値です。(iii) 変数の値は型 item* である必要があります。

サンプル

```
<xsl:variable name="xpath" select="'$p3, $p2, $p1'" />
<xsl:value-of select="altova:evaluate($xpath, 10, 20, 'hi')" />
```

outputs "hi 20 10"

上のリストに関して、以下の点に注意してください：

- altova:evaluate 式の第 2 引数は、変数 $p1 に割り当てられた値で、第三の引数は変数 $p2 に割り当てられた値です。
- 関数の第 4 番目の引数は、引用符による囲いで表示された文字列の値です。
- xs:variable 要素の select 属性は、XPath 式を提供します。この式は xs:string の型である必要があり、一重引用符で囲まれています。

変数の使用方法を更に説明するサンプル

- ```
  <xsl:variable name="xpath" select="'$p1'" />
  <xsl:value-of select="altova:evaluate($xpath, //Name[1])" />
  ```
  最初の Name *要素の出力値*

- ```
  <xsl:variable name="xpath" select="'$p1'" />
  <xsl:value-of select="altova:evaluate($xpath, '//Name[1]')" />
  ```
  Outputs "//Name[1]"

altova:evaluate() 拡張関数は、XSLT スタイルシート内の XPath 式が動的に評価される必要のあるシチュエーションで役に立ちます。例えば、ユーザーが並べ替えの必要条件をリクエストする場合、このシチュエーションは属性 UserReq/@sortkey に保管されます。スタイルシートでは、以下の式が使用できます: <xsl:sort select="**altova:evaluate(../UserReq/@sortkey)**" order="ascending"/>。altova:evaluate() 関数は、コンテキストノードの親の UserReq 子要素の sortkey 属性を読み込みます。sortkey 属性の値が Price の場合、Price は、altova:evaluate() 関数により返され、select 属性 <xsl:sort select="**Price**" order="ascending"/> の値になります。この sort 命令が、Order という要素のコンテキスト内で発生する場合、Order 要素は Price の子の値に従い並べ替えられます。また、@sortkey の値が、Date の場合、Order 要素は、Date の子の値に従い並べ替えられます。ですから、Order の並べ替えの条件は、ランタイムでの sortkey 属性から選択されます。これは、以下の式などでは達成することはできません: <xsl:sort select="**../UserReq/@sortkey**" order="ascending"/>。上の場合、並べ替え条件は sortkey 属性自身であり、Price または Date（*または、現在の* sortkey *のコンテンツ*）ではありません。

メモ 静的なコンテキストは、変数以外以外で、呼び出し環境の名前空間、型、機能、を含みます。ベース URI とデフォルトの名前空間は継承されます。

追加サンプル

- 静的な変数: <xsl:value-of select="$i3, $i2, $i1" />
  3 つの変数の値を出力します。

- 動的な変数を持つ動的 XPath 式:
```
<xsl:variable name="xpath" select="'$p3, $p2, $p1'" />
<xsl:value-of select="altova:evaluate($xpath, 10, 20, 30)" />
```
"30 20 10" を出力します。

- 動的な変数を持たない XPath 式:
```
<xsl:variable name="xpath" select="'$p3, $p2, $p1'" />
<xsl:value-of select="altova:evaluate($xpath)" />
```
出力エラー: $p3 に対して定義されている変数はありません。

encode-for-rtf [altova:]

altova:encode-for-rtf(input *as xs:string*, **preserveallwhitespace** *as xs:boolean*, **preservenewlines** *as xs:boolean*) を **xs:string** とする XSLT2 XSLT3

RTF のためのコードに入力文字列を変換します。空白と新しい行は、それぞれの引数により指定される boolean の値に基づき保管されます。

[トップ 1490]

XBRL 関数

Altova XBRL 関数は XBRL をサポートする Altova 製品のエディションのみで使用することができます。

xbrl-footnotes [altova:]

altova:xbrl-footnotes(*node()***)** を **node()*** とする XSLT2 XSLT3

ノードを入力引数として必要とし、入力ノードに参照される XBRL フットノート ノードを返します。

xbrl-labels [altova:]

altova:xbrl-labels(*xs:QName*, *xs:string***)** を **node()*** とする XSLT2 XSLT3

以下の 2 つの入力引数を必要とします: ノード名とノードを含むタクソノミファイルロケーション。関数は、入力ノードと関連した XBRL ラベルノードを返します。

[トップ 1490]

26.2.1.2 XPath/XQuery 関数: 日付と時刻

Altova の日付/時刻拡張関数は XPath と XQuery 式で使用することができ、XML スキーマの異なる日付および時刻データ型で保存されているデータを処理するための追加機能を提供します。このセクションの関数は、Altova の **XPath 3.0** および **XQuery 3.0** エンジンで使用することができます。これらの関数は、XPath/XQuery コンテキストで使用することができます。

関数の名前指定と言語の適用性に関するメモ

Altova 拡張関数はXPath/XQuery 式で使用することができ、XPath、XQuery、およびXSLT 関数の標準ライブラリで使用可能な機能に更なる機能性を与えます。Altova 拡張関数はAltova **拡張関数名前空間**、**http://www.altova.com/xslt-extensions** に収められており、**altova:** プレフィックスが、このセクションでは使用されます。製品の今後のバージョンが拡張機能への継続的サポート、または個別の関数の振る舞いは変更する可能性があることに注意してください。Altova 拡張機能へのサポートに関しては、今後のリリースのドキュメントを参照してください。

XPath 関数 (XSLT 内のXPath 式で使用):	XP1 XP2 XP3.1
XSLT 関数 (XSLT 内のXPath 式で使用):	XSLT1 XSLT2 XSLT3
XQuery 関数 (XQuery 内のXQuery 式で使用):	XQ1 XQ3.1

機能によりグループ化

- xs:dateTime に期間を追加して、xs:dateTimeを返す 1494
- xs:date に期間を追加して、xs:date を返す 1496
- xs:time に期間を追加して、return xs:time を返す 1497
- フォーマットと期間の取得 1496
- 現在の日付/時刻を生成する関数からタイムゾーンを削除する 1498
- 期間から日付、時刻、分数、および、秒数を返す 1499
- 日付から整数を週の曜日として返す 1501
- 日付から週数を整数として返す 1501
- 各型の構文コンポーネントから日付、時刻、期間 の型を構築する 1503
- 文字列入力から 日付、日付時刻 または 時刻 を構築する 1504
- 年齢に関連した関数 1506

アルファベット順にグループ化

altova:add-days-to-date 1496
altova:add-days-to-dateTime 1494
altova:add-hours-to-dateTime 1494
altova:add-hours-to-time 1497
altova:add-minutes-to-dateTime 1494
altova:add-minutes-to-time 1497
altova:add-months-to-date 1496
altova:add-months-to-dateTime 1494
altova:add-seconds-to-dateTime 1494
altova:add-seconds-to-time 1497
altova:add-years-to-date 1496
altova:add-years-to-dateTime 1494
altova:age 1506
altova:age-details 1506
altova:build-date 1503
altova:build-duration 1503
altova:build-time 1503
altova:current-dateTime-no-TZ 1498
altova:current-date-no-TZ 1498
altova:current-time-no-TZ 1498
altova:date-no-TZ 1498
altova:dateTime-no-TZ 1498
altova:days-in-month 1499
altova:hours-from-dateTimeDuration-accumulated 1499
altova:minutes-from-dateTimeDuration-accumulated 1499
altova:seconds-from-dateTimeDuration-accumulated 1499

altova:format-duration 1496
altova:parse-date 1504
altova:parse-dateTime 1504
altova:parse-duration 1496
altova:parse-time 1504
altova:time-no-TZ 1498
altova:weekday-from-date 1501
altova:weekday-from-dateTime 1501
altova:weeknumber-from-date 1502
altova:weeknumber-from-dateTime 1502

[トップ 1492]

xs:dateTime に期間を追加する XP3.1 XQ3.1

これらの関数は**xs:dateTime** に期間を追加し、**xs:dateTime** を返します。xs:dateTime 型はCCYY-MM-DDThh:mm:ss.sss のフォーマットです。これはxs:date とxs:time フォーマットの連結で、T により区切られています。タイムゾーンサフィックス+01:00 (for example)は任意です。

add-years-to-dateTime [altova:]

altova:add-years-to-dateTime(DateTime *as xs:dateTime*, **Years** を *xs:integer*)
asxs:dateTime とする **XP3.1** **XQ3.1**
日付までの期間を年数で表示します。第 2 の引数は第 1 の引数として与えられたxs:date に追加される年数です。結果はxs:date 型です。

サンプル

- **altova:add-years-to-dateTime**(xs:dateTime("2014-01-15T14:00:00"), 10) は2024-01-15T14:00:00 を返します。
- **altova:add-years-to-dateTime**(xs:dateTime("2014-01-15T14:00:00"), -4) は2010-01-15T14:00:00 を返します。

add-months-to-dateTime [altova:]

altova:add-months-to-dateTime(DateTime *as xs:dateTime*, **Months** を *xs:integer*)
asxs:dateTime とする **XP3.1** **XQ3.1**
xs:dateTime に月数での期間を追加します（*下のサンプル参照*）。第 2 の引数は、第 1 の引数として与えられたxs:dateTime に追加される月数です。結果はxs:dateTime 型です。

サンプル

- **altova:add-months-to-dateTime**(xs:dateTime("2014-01-15T14:00:00"), 10) 2014-11-15T14:00:00 を返します。
- **altova:add-months-to-dateTime**(xs:dateTime("2014-01-15T14:00:00"), -2) 2013-11-15T14:00:00 を返します。

add-days-to-dateTime [altova:]

altova:add-days-to-dateTime(DateTime *as xs:dateTime*, **Days** *as xs:integer*) を**xs:dateTime** とする **XP3.1** **XQ3.1**
xs:dateTime に日数での期間を追加します（*下のサンプル参照*）。第 2 の引数は、第 1 の引数として与えられたxs:dateTime に追加される日数です。結果はxs:dateTime 型です。

(C) 2015-2021 Altova GmbH

サンプル

- **altova:add-days-to-dateTime**(xs:dateTime("2014-01-15T14:00:00"), 10) は 2014-01-25T14:00:00 を返します。
- **altova:add-days-to-dateTime**(xs:dateTime("2014-01-15T14:00:00"), -8) は 2014-01-25T14:00:00 を返します。

add-hours-to-dateTime [altova:]

altova:add-hours-to-dateTime(DateTime *as xs:dateTime*, **Hours** *as xs:integer*) を **xs:dateTime** とする **XP3.1** **XQ3.1**

xs:dateTime に時間数での期間を追加します（*下のサンプル参照*）。第 2 の引数は、第 1 の引数として与えられた xs:dateTime に追加される時間数です。結果は xs:dateTime 型です。

サンプル

- **altova:add-hours-to-dateTime**(xs:dateTime("2014-01-15T13:00:00"), 10) は 2014-01-15T23:00:00 を返します。
- **altova:add-hours-to-dateTime**(xs:dateTime("2014-01-15T13:00:00"), -8) は 2014-01-15T05:00:00 を返します。

add-minutes-to-dateTime [altova:]

altova:add-minutes-to-dateTime(DateTime *as xs:dateTime*, **Minutes** *as xs:integer*) を **xs:dateTime** とする **XP3.1** **XQ3.1**

xs:dateTime に分数での期間を追加します（*下のサンプル参照*）。第 2 の引数は、第 1 の引数として与えられた xs:dateTime に追加される分数です。結果は xs:dateTime 型です。

サンプル

- **altova:add-minutes-to-dateTime**(xs:dateTime("2014-01-15T14:10:00"), 45) 2014-01-15T14:55:00 を返します。
- **altova:add-minutes-to-dateTime**(xs:dateTime("2014-01-15T14:10:00"), -5) 2014-01-15T14:05:00 を返します。

add-seconds-to-dateTime [altova:]

altova:add-seconds-to-dateTime(DateTime *as xs:dateTime*, **Seconds** を *xs:integer*) **asxs:dateTime** とする **XP3.1** **XQ3.1**

xs:dateTime に秒数での期間を追加します（*下のサンプル参照*）。第 2 の引数は、第 1 の引数として与えられた xs:dateTime に追加される秒数です。結果は xs:dateTime 型です。

サンプル

- **altova:add-seconds-to-dateTime**(xs:dateTime("2014-01-15T14:00:10"), 20) 2014-01-15T14:00:30 を返します。
- **altova:add-seconds-to-dateTime**(xs:dateTime("2014-01-15T14:00:10"), -5) 2014-01-15T14:00:05 を返します。

[トップ 1492]

xs:date に期間を追加する XP3.1 XQ3.1

これらの関数は**xs:date** に期間を追加し、**xs:date** を返します。xs:date 型はCCYY-MM-DD フォーマットです。

add-years-to-date [altova:]

altova:add-years-to-date(Date *as xs:date*, **Years** *as xs:integer*) を**xs:date** とする XP3.1 XQ3.1

日付までの期間を年数で表示します。第 2 の引数は第 1 の引数として与えられたxs:date に追加される年数です。結果は xs:date 型です。

サンプル

- **altova:add-years-to-date**(xs:date("2014-01-15"), 10) は2024-01-15 を返します。
- **altova:add-years-to-date**(xs:date("2014-01-15"), -4) は2010-01-15 を返します。

add-months-to-date [altova:]

altova:add-months-to-date(Date *as xs:date*, **Months** *as xs:integer*) を**xs:date** とする XP3.1 XQ3.1

日付までの期間を月数で表示します。第 2 の引数は第 1 の引数として与えられたxs:date に追加される月数です。結果は xs:date 型です。

サンプル

- **altova:add-months-to-date**(xs:date("2014-01-15"), 10) 2014-11-15 を返します。
- **altova:add-months-to-date**(xs:date("2014-01-15"), -2) 2013-11-15 を返します。

add-days-to-date [altova:]

altova:add-days-to-date(Date *as xs:date*, **Days** *as xs:integer*) を**xs:date** とする XP3.1 XQ3.1

日付までの期間を日数で表示します。第 2 の引数は第 1 の引数として与えられたxs:date に追加される日数です。結果は xs:date 型です。

サンプル

- **altova:add-days-to-date**(xs:date("2014-01-15"), 10) は 2014-01-25 を返します。
- **altova:add-days-to-date**(xs:date("2014-01-15"), -8) は2014-01-07 を返します。

[トップ 1492]

フォーマットと期間の取得 XP3.1 XQ3.1

これらの関数は入力 **xs:duration** または**xs:string** を解析し、それぞれan **xs:string** または**xs:duration** を返します。

format-duration [altova:]

altova:format-duration(Duration *as xs:duration*, **Picture** *as xs:string*) **asxs:string** とする XP3.1 XQ3.1

第 1 の引数として提出された期間を、第 2 の引数として提出された文字列によりをフォーマットします。出力は、文字列によりフォーマットされたテキスト文字列です。

サンプル

- **altova:format-duration**(xs:duration("P2DT2H53M11.7S"), "Days:[D01] Hours:[H01] Minutes:[m01] Seconds:[s01] Fractions:[f0]") は"Days:02 Hours:02 Minutes:53

(C) 2015-2021 Altova GmbH

Seconds:11 Fractions:7" を返します。

- **altova:format-duration**(xs:duration("P3M2DT2H53M11.7S"), "Months:[M01] Days:[D01] Hours:[H01] Minutes:[m01]") は "Months:03 Days:02 Hours:02 Minutes:53" を返します。

parse-duration [altova:]

altova:parse-duration(InputString *as xs:string*, **Picture** *as xs:string*) を **xs:duration** とする **XP3.1** **XQ3.1**

パターン化された文字列を最初の引数として、文字を第 2 の引数とします。入力文字列は文字をベースに解析され、xs:duration が返されます。

サンプル

- **altova:parse-duration**("Days:02 Hours:02 Minutes:53 Seconds:11 Fractions:7"), "Days:[D01] Hours:[H01] Minutes:[m01] Seconds:[s01] Fractions:[f0]") は "P2DT2H53M11.7S" を返します。
- **altova:parse-duration**("Months:03 Days:02 Hours:02 Minutes:53 Seconds:11 Fractions:7", "Months:[M01] Days:[D01] Hours:[H01] Minutes:[m01]") は "P3M2DT2H53M" を返します。

[トップ 1492]

xs:time に期間を追加する **XP3.1** **XQ3.1**

これらの関数は **xs:time** に期間を追加し、 **xs:time** を返します。 **xs:time** 型は hh:mm:ss.sss 構文フォームです。文字 Z は協定世界時（UTC）を表します。他のタイムゾーンは、UTC との差異を +hh:mm または -hh:mm のフォーマットで表示しています。タイムゾーンの値が無い場合は、UTC ではない未知のタイムゾーンとして見なされます。

add-hours-to-time [altova:]

altova:add-hours-to-time(Time *as xs:time*, **Hours** *as xs:integer*) を **xs:time** とする **XP3.1** **XQ3.1**

日付までの期間を時間数で表示します。第 2 の引数は第 1 の引数として与えられた xs:time に追加される時間数です。結果は xs:time 型です。

サンプル

- **altova:add-hours-to-time**(xs:time("11:00:00"), 10) は 21:00:00 を返します。
- **altova:add-hours-to-time**(xs:time("11:00:00"), -7) は 04:00:00 を返します。

add-minutes-to-time [altova:]

altova:add-minutes-to-time(Time *as xs:time*, **Minutes** *as xs:integer*) を **xs:time** とする **XP3.1** **XQ3.1**

日付までの期間を分数で表示します。第 2 の引数は第 1 の引数として与えられた xs:date に追加される分数です。結果は xs:date 型です。

サンプル

- **altova:add-minutes-to-time**(xs:time("14:10:00"), 45) 14:55:00 を返します。
- **altova:add-minutes-to-time**(xs:time("14:10:00"), -5) 14:05:00 を返します。

add-seconds-to-time [altova:]

altova:add-seconds-to-time(Time *as xs:time*, **Minutes** *as xs:integer*) を **xs:time** とする **XP3.1** **XQ3.1**

時間までの期間を秒数で表示します。第 2 の引数は、第 1 の引数として与えられた xs:time に追加される秒数です。結果は xs:time 型です。第 2 のコンポーネントは 0 から 59.999 の範囲であることができます。

サンプル

- **altova:add-seconds-to-time**(xs:time("14:00:00"), 20) は 14:00:20 を返します。
- **altova:add-seconds-to-time**(xs:time("14:00:00"), 20.895) は 14:00:20.895 を返します。

[トップ 1492]

日付/時刻データ型からタイムゾーンの部分を削除する **XP3.1** **XQ3.1**

これらの関数は、現在の **xs:dateTime**、**xs:date**、または **xs:time** 値からそれぞれタイムゾーンを削除します。xs:dateTime と xs:dateTimeStamp の差異は、後者のタイムゾーンが必要な場合です。(前者の場合は任意です。) xs:dateTimeStamp 値のフォーマットは CCYY-MM-DDThh:mm:ss.sss±hh:mm または CCYY-MM-DDThh:mm:ss.sssZ です。、日付と時刻が xs:dateTimeStamp としてシステムクロックから読み込まれる場合、 current-dateTime-no-TZ() 関数がタイムゾーンを削除するために使用されます。

current-date-no-TZ [altova:]

altova:current-date-no-TZ() を **xs:date** とする **XP3.1** **XQ3.1**

この関数は引数を必要としません。current-date() (システムクロックによる現在の時刻) のタイムゾーンの部分を削除し、xs:date の値を返します。

サンプル

現在の date が **2014-01-15+01:00** の場合:

- **altova:current-date-no-TZ**() は 2014-01-15 を返します。

current-dateTime-no-TZ [altova:]

altova:current-dateTime-no-TZ() を **xs:dateTime** とする **XP3.1** **XQ3.1**

この関数は引数を必要としません。current-dateTime() (システムクロックによる現在の時刻) のタイムゾーンの部分を削除し、xs:dateTime の値を返します。

サンプル

現在の dateTime が **2014-01-15T14:00:00+01:00** の場合:

- **altova:current-dateTime-no-TZ**() は 2014-01-15T14:00:00 を返します。

current-time-no-TZ [altova:]

altova:current-time-no-TZ() を **xs:time** とする **XP3.1** **XQ3.1**

この関数は引数を必要としません。current-time() (システムクロックによる現在の時刻) のタイムゾーンの部分を削除し、xs:time の値を返します。

サンプル

現在の time が **14:00:00+01:00** の場合:

(C) 2015-2021 Altova GmbH

- **altova:current-time-no-TZ**() は14:00:00 を返します。

date-no-TZ [altova:]

altova:date-no-TZ(InputDate *as xs:date*) を**xs:date** とする **XP3.1** **XQ3.1**
この関数はxs:date 引数を必要とし、タイムゾーンの部分を削除し、xs:date の値を返します。日付が変更されていない点に注意してください。

サンプル

- **altova:date-no-TZ**(xs:date("2014-01-15+01:00")) は2014-01-15 を返します。

dateTime-no-TZ [altova:]

altova:dateTime-no-TZ(InputDateTime *as xs:dateTime*) を**xs:dateTime** とする **XP3.1** **XQ3.1**
この関数はxs:dateTime 引数を必要とし、タイムゾーンの部分を削除し、xs:dateTime の値を返します。日付が変更されていない点に注意してください。

サンプル

- **altova:dateTime-no-TZ**(xs:date("2014-01-15T14:00:00+01:00")) は2014-01-15T14:00:00 を返します。

time-no-TZ [altova:]

altova:time-no-TZ(InputTime *as xs:time*) を**xs:time** とする **XP3.1** **XQ3.1**
この関数はxs:time 引数を必要とし、タイムゾーンの部分を削除し、xs:time 値を返します。時刻は変更されていないことに注意してください。

サンプル

- **altova:time-no-TZ**(xs:time("14:00:00+01:00")) は14:00:00 を返します。

[トップ 1492]

期間から日数、時間数、分数、および、秒数を返す **XP3.1** **XQ3.1**

これらの関数は、期間から、月内の日数、時間数、分数、秒数をそれぞれ返します。

days-in-month [altova:]

altova:days-in-month(Year *as xs:integer*, **Month** *as xs:integer*) **asxs:integer** **XP3.1** **XQ3.1**
は指定された月内の日数 を返します。Year とMonth 引数を使用して月を指定することができます。

サンプル

- **altova:days-in-month**(2018, 10) は31 を返します。
- **altova:days-in-month**(2018, 2) は28 を返します。
- **altova:days-in-month**(2020, 2) は29 を返します。

hours-from-dayTimeDuration-accumulated

altova:hours-from-dayTimeDuration-accumulated(DayAndTime *as xs:duration*) **asxs:integer** **XP3.1** **XQ3.1**

DayAndTime 引数 (which is of type xs:duration) により提供される期間内の時間の総数を返します。Day と Time コンポーネント内の時間は、整数である結果に追加されます。60 分として新規の一時間は考えられます。ネガティブな時間の値で期間の結果をナビゲートすることができます。

サンプル

- **altova:hours-from-dayTimeDuration-accumulated**(xs:duration("P5D") は5日間の時間の総計である 120 を返します。
- **altova:hours-from-dayTimeDuration-accumulated**(xs:duration("P5DT2H") は5日間の時間の総計プラス2時間である 122 を返します。
- **altova:hours-from-dayTimeDuration-accumulated**(xs:duration("P5DT2H60M") は5日間の時間の総計プラス2時間プラス60分である 123 を返します。
- **altova:hours-from-dayTimeDuration-accumulated**(xs:duration("P5DT2H119M") は5日間の時間の総計プラス2時間とプラス119分である 123 を返します。
- **altova:hours-from-dayTimeDuration-accumulated**(xs:duration("P5DT2H120M") は5日間の時間の総計プラス2時間プラス120分である 124 を返します。
- **altova:hours-from-dayTimeDuration-accumulated**(xs:duration("-P5DT2H") は -122 を返します。

minutes-from-dayTimeDuration-accumulated

altova:minutes-from-dayTimeDuration-accumulated(DayAndTime *as xs:duration*)
asxs:integer **XP3.1** **XQ3.1**

DayAndTime 引数 (which is of type xs:duration) により提出される期間内の分数の総数を返します Day と Time コンポーネントは、追加され、整数である結果に追加されます。ネガティブなマイナスの値で期間の結果をナビゲートします。

サンプル

- **altova:minutes-from-dayTimeDuration-accumulated**(xs:duration("PT60M") は を返します。60
- **altova:minutes-from-dayTimeDuration-accumulated**(xs:duration("PT1H") は一時間内の分数である 60 を返します。
- **altova:minutes-from-dayTimeDuration-accumulated**(xs:duration("PT1H40M") は 100 を返します。
- **altova:minutes-from-dayTimeDuration-accumulated**(xs:duration("P1D") は一日の秒数の総数である 1440 を返します。
- **altova:minutes-from-dayTimeDuration-accumulated**(xs:duration("-P1DT60M") は を返します。-1500

seconds-from-dayTimeDuration-accumulated

altova:seconds-from-dayTimeDuration-accumulated(DayAndTime *as xs:duration*)
asxs:integer **XP3.1** **XQ3.1**

(xs:duration の型である)DayAndTime 引数 により提出された期間内の秒数の総数を返します。Day と Time コンポーネント内の秒数は、整数である結果に追加されます。ネガティブな秒の値で期間をナビゲートします。

サンプル

- **altova:seconds-from-dayTimeDuration-accumulated**(xs:duration("PT1M") は一分内の秒数である 60 を返します。
- **altova:seconds-from-dayTimeDuration-accumulated**(xs:duration("PT1H") は一時間内の秒数の総数である 3600 を返します。
- **altova:seconds-from-dayTimeDuration-accumulated**(xs:duration("PT1H2M") は 3720 を返します。

- **altova:seconds-from-dayTimeDuration-accumulated**(xs:duration("P1D") は一日の秒数の総数である 86400 を返します。
- **altova:seconds-from-dayTimeDuration-accumulated**(xs:duration("-P1DT1M") は -86460 を返します。

xs:dateTime または xs:dateReturn から週数を返す XP3.1 XQ3.1

これらの関数は、xs:dateTime または xs:date から曜日を(整数として)を返します。曜日は(米国式フォーマットを使用して) 1 から 7 と番号づけられています。 日曜=1 と番号付けられます。欧州のフォーマットでは月曜 (=1) として番号付けられます。日曜=1 である米国フォーマットは整数 0 がフォーマットを表示するために使用できる箇所で設定することができます。

weekday-from-dateTime [altova:]

altova:weekday-from-dateTime(DateTime *as xs:dateTime*) を **xs:integer** とする XP3.1 XQ3.1

時刻付きの日付を単一の引数として、この日付の曜日を正数として返します。曜日は 日曜=1 から開始して番号を付けます。(月曜=1 とする) ヨーロッパのフォーマットが必要な場合、この関数の他の署名を使用します (*下の次の署名を参照*)。

サンプル

- **altova:weekday-from-dateTime**(xs:dateTime("2014-02-03T09:00:00")) は月曜を表示する 2 を返します。

altova:weekday-from-dateTime(DateTime *as xs:dateTime*, **Format** *as xs:integer*) **asxs:integer** とする XP3.1 XQ3.1

時間月の日付を最初の引数として、この日付の曜日を正数として返します。曜日は 月曜=1 から開始して番号を付けます。第 2 の引数(整数)が 0 の場合、日曜=1 から開始して、曜日は 1 から 7 と番号を付けます。第 2 の引数が 0 以外の整数の場合、月曜=1 です。第 2 の引数がない場合、関数はこの関数の他の署名を持つと読み込まれます (*前の次の署名を参照*)。

サンプル

- **altova:weekday-from-dateTime**(xs:dateTime("2014-02-03T09:00:00"), 1) は月曜を表示する 1 を返します。
- **altova:weekday-from-dateTime**(xs:dateTime("2014-02-03T09:00:00"), 4) は月曜を表示する 1 を返します。
- **altova:weekday-from-dateTime**(xs:dateTime("2014-02-03T09:00:00"), 0) は月曜を表示する 2 を返します。

weekday-from-date [altova:]

altova:weekday-from-date(Date *as xs:date*) を **xs:integer** とする XP3.1 XQ3.1

日付を単一の引数として、この日付の曜日を正数として返します。曜日は 日曜=1 から開始して番号を付けます。(月曜=1 とする) ヨーロッパのフォーマットが必要な場合、この関数の他の署名を使用します (*下の次の署名を参照*)。

サンプル

- **altova:weekday-from-date**(xs:date("2014-02-03+01:00")) は月曜を表示する 2 を返します。

altova:weekday-from-date(Date *as xs:date*, **Format** *as xs:integer*) を **xs:integer** とする XP3.1 XQ3.1

日付を最初の引数とし、この日付の曜日を正数として返します。曜日は 月曜=1 から開始して番号を付けます。第 2 (フォーマット) 引数が 0, の場合、日曜=1 から開始し、曜日は 1 から 7 で番号付けられます。第 2 引数が整数で 0 以外の場合、月曜=1 です。第 2 の引数がない場合、関数はこの関数の他の署名を持つと読み込まれます (*前の署名を参照*)。

サンプル

- **altova:weekday-from-date**(xs:date("2014-02-03"), 1) は月曜を示す 1 を返します。
- **altova:weekday-from-date**(xs:date("2014-02-03"), 4) は月曜を示す 1 を返します。
- **altova:weekday-from-date**(xs:date("2014-02-03"), 0) は月曜を示す 2 を返します。

[トップ 1492]

xs:dateTime または xs:date から週数を返す XP2 XQ1 XP3.1 XQ3.1

これらの関数は週数(整数として)を xs:dateTime から xs:date. から返します。週の番号付けは、米国、欧州、イスラムのカレンダーフォーマットで使用することができます。週の始まりが異なるため、週数の番号付けは、カレンダーのフォーマットにより異なります。(米国フォーマットでは、日曜、欧州フォーマットでは月曜、イスラムフォーマットでは土曜が週の開始日です)。

weeknumber-from-date [altova:]

altova:weeknumber-from-date(Date *as xs:date*, **Calendar** *as xs:integer*) を **xs:integer** とする XP2 XQ1 XP3.1 XQ3.1

正数として提出された Date 引数の週数を返します。第 2 の引数(カレンダー)は続くカレンダーシステムを指定します。
サポートされるカレンダー の値は次のとおりです:

- 0 = US 米国のカレンダー(週の始まりは日曜日)
- 1 = ISO 標準、 欧州のカレンダー(週の始まりは月曜日)
- 2 = イスラムのカレンダー(週の始まりは土曜日)

デフォルトは 0 です。

サンプル

- **altova:weeknumber-from-date**(xs:date("2014-03-23"), 0) は 13 を返します。
- **altova:weeknumber-from-date**(xs:date("2014-03-23"), 1) は 13 を返します。
- **altova:weeknumber-from-date**(xs:date("2014-03-23"), 2) は 13 を返します。
- **altova:weeknumber-from-date**(xs:date("2014-03-23")) は 13 を返します。

上のサンプル(2014-03-23) の date の曜日は日曜日です。米国およびイスラムのカレンダーは欧州カレンダーのこの日付よりも一週間先です。

weeknumber-from-dateTime [altova:]

altova:weeknumber-from-dateTime(DateTime *as xs:dateTime*, **Calendar** *as xs:integer*) を **xs:integer** とする XP2 XQ1 XP3.1 XQ3.1

正数として提出された DateTime 引数の週数を返します。第 2 の引数(カレンダー)は続くカレンダーシステムを指定します。
サポートされるカレンダー の値は次のとおりです:

- 0 = US 米国のカレンダー(週の始まりは日曜日)
- 1 = ISO 標準、 欧州のカレンダー(週の始まりは月曜日)
- 2 = イスラムのカレンダー(週の始まりは土曜日)

Default is 0.

サンプル

- **altova:weeknumber-from-dateTime**(xs:dateTime("2014-03-23T00:00:00"), 0) は13 を返します。
- **altova:weeknumber-from-dateTime**(xs:dateTime("2014-03-23T00:00:00"), 1) は13 を返します。
- **altova:weeknumber-from-dateTime**(xs:dateTime("2014-03-23T00:00:00"), 2) は13 を返します。
- **altova:weeknumber-from-dateTime**(xs:dateTime("2014-03-23T00:00:00")) は13 を返します。

上のサンプル(2014-03-23T00:00:00) のdateTime の曜日は日曜日です。米国およびイスラムのカレンダーは欧州カレンダーのこの日付より先一週間先です。

[トップ 1492]

各型の構文コンポーネントから日付、時刻、期間の型を構築する XP3.1 XQ3.1

関数は、xs:date, xs:time または xs:duration の構文コンポーネントを入力引数とし、引数を結合させて対応するデータ型を構築します。

build-date [altova:]

altova:build-date(Year *as xs:integer*, **Month** *as xs:integer*, **Date** *as xs:integer*) を **xs:date** とする XP3.1 XQ3.1

第 1 、第 2 、第3 の引数は、それぞれ、年、月、日を表します。xs:date 型の値を構築するため結合されます。整数の値は、特定の日付の一部の適正な範囲内である必要があります。例えば第 2 の引数 (月の部分) は12 以上であってはなりません。

サンプル

- **altova:build-date**(2014, 2, 03) は2014-02-03 を返します。

build-time [altova:]

altova:build-time(Hours *as xs:integer*, **Minutes** *as xs:integer*, **Seconds** *as xs:integer*) を**xs:time** とする XP3.1 XQ3.1

第 1 、第 2 、第3 引数はそれぞれ時間数 (0 から23)、分数 (0 から59)、および秒数 (0 から59) の値です。これらの値は、xs:time 型の値を作成するために結合されます。整数の値は、それぞれの部分の正しい範囲内である必要があります。例えば、秒 (分) 引数は、59 以上であってはなりません。値にタイムゾーンを追加するには、この関数の他の署名を使用してください (*次の署名を参照*)。

サンプル

- **altova:build-time**(23, 4, 57) は23:04:57 を返します。

altova:build-time(Hours *as xs:integer*, **Minutes** *as xs:integer*, **Seconds** *as xs:integer*, **TimeZone** *as xs:string*) を**xs:time** とする XP3.1 XQ3.1

第 1 、第 2 、第 3 引数はそれぞれ時間数 (0 から23)、分数 (0 から59)、および秒数 (0 から59) の値です。第四の引数は、値の一部としてタイムゾーンを与えます。4 つの引数が結合され、xs:time 型の値を構築します。例えば、第 2 の引数 (分数) は59 以上であってはなりません。

サンプル

- **altova:build-time**(23, 4, 57, '+1') は23:04:57+01:00 を返します。

build-duration [altova:]

altova:build-duration(Years *as xs:integer*, **Months** *as xs:integer*) を **xs:yearMonthDuration** とする **XP3.1** **XQ3.1**

xs:yearMonthDuration 型の値を構築するためには 2 つの引数が必要です。最初の引数は期間の 年数 値の部分を与え、第 2 の引数は、月数 値の部分を与えます。第 2 の引数（月数）が同じまたはより大きくなると、整数は 12 により分割されます。商は、年数 の部分を表す 最初の引数に加算され、（除算の）残りはの部分は 月数の部分を表すために使用されます。期間の xs:dayTimeDuration 型を作成するには、次の署名を参照してください。

サンプル

- **altova:build-duration**(2, 10) は P2Y10M を返します。
- **altova:build-duration**(14, 27) は P16Y3M を返します。
- **altova:build-duration**(2, 24) は P4Y を返します。

altova:build-duration(Days *as xs:integer*, **Hours** *as xs:integer*, **Minutes** *as xs:integer*, **Seconds** *as xs:integer*) を **xs:dayTimeDuration** とする **XP3.1** **XQ3.1**

xs:dayTimeDuration 型の値を構築するためには 4 つの引数が必要です。最初の引数は期間の 日数 値の部分で、第 2 、第 3 、第 4 引数は、それぞれ期間の時間数、分数、秒数 値のです。これらの 3 つの時間 引数は、次の高い単位の値に加算され、結果は期間の全体の値を計算するために使用されます。例えば、72 秒は、1M+12S（1 分 と 12 秒）に変換され、この値が期間全体の値を計算するために使用されます。xs:yearMonthDuration 型の期間を構築するためには、次の署名を参照してください。

サンプル

- **altova:build-duration**(2, 10, 3, 56) は P2DT10H3M56S を返します。
- **altova:build-duration**(1, 0, 100, 0) は P1DT1H40M を返します。
- **altova:build-duration**(1, 0, 0, 3600) は P1DT1H を返します。

[トップ 1492]

文字列入力から 日付、日付時刻 または 時刻 を構築する **XP2** **XQ1** **XP3.1** **XQ3.1**

関数は、文字列を引数として、xs:date、xs:dateTime または xs:time データ型を構築します。文字列は、データ型のコンポーネントを提出されたパターン引数をベースにして分析されます。

parse-date [altova:]

altova:parse-date(Date *as xs:string*, **DatePattern** *as xs:string*) を **xs:date** とする **XP2** **XQ1** **XP3.1** **XQ3.1**

Date 入力文字列を **xs:date** の値として返します。第二の引数である **DatePattern** は、入力文字列のパターン（コンポーネントのシーケンス）を指定します。**DatePattern** は、下にリストされるコンポーネント指定子および任意の文字であるコンポーネントセパレータと共に説明されています。下のサンプルを参照してください。

D 日付
M 月
Y 年

DatePattern のパターンは、**Date** のパターンに一致する必要があります。出力は **xs:date** 型であるため、出力は常に **YYYY-MM-DD** 構文フォーマットになります。

サンプル

- **altova:parse-date**(xs:string("06-03-2014"), "[D]-[M]-[Y]") は 2014-03-06 を返します。
- **altova:parse-date**(xs:string("06-03-2014"), "[M]-[D]-[Y]") は 2014-03-06 を返します。
- **altova:parse-date**("06/03/2014", "[M]/[D]/[Y]") は 2014-03-06 を返します。
- **altova:parse-date**("06 03 2014", "[M] [D] [Y]") は 2014-03-06 を返します。

(C) 2015-2021 Altova GmbH

- **altova:parse-date**("6 3 2014", "[M] [D] [Y]") は2014-03-06 を返します。

parse-dateTime [altova:]

altova:parse-dateTime(DateTime *as xs:string*, **DateTimePattern** *as xs:string*) を **xs:dateTime** とする **XP2** **XQ1** **XP3.1** **XQ3.1**

DateTime 入力文字列を**xs:dateTime** の値として返します。第二の引数である**DateTimePattern** は、入力文字列のパターン（コンポーネントのシーケンス）を指定します。**DateTimePattern** は、下にリストされるコンポーネント指定子および任意の文字であるコンポーネント セパレータと共に説明されています。下のサンプルを参照してください。

D	日
M	月
Y	年
H	時間
m	分
s	秒

DateTimePattern のパターンは、**DateTime** のパターンに一致する必要があります。出力は**xs:dateTime** 型であるため、出力は常に**YYYY-MM-DDTHH:mm:ss** 構文フォーマットになります。

サンプル

- **altova:parse-dateTime**(xs:string("06-03-2014 13:56:24"), "[D]-[M]-[Y] [H]:[m]:[s]") は2014-03-06T13:56:24 を返します。
- **altova:parse-dateTime**("time=13:56:24; date=06-03-2014", "time=[H]:[m]:[s]; date=[D]-[M]-[Y]") は2014-03-06T13:56:24 を返します。

parse-time [altova:]

altova:parse-time(Time *as xs:string*, **TimePattern** *as xs:string*) を**xs:time** とする **XP2** **XQ1** **XP3.1** **XQ3.1**

Time 入力文字列を**xs:time** の値として返します。第二の引数である**TimePattern** は、入力文字列のパターン（コンポーネントのシーケンス）を指定します。**TimePattern** は、下にリストされるコンポーネント指定子および任意の文字であるコンポーネント セパレータと共に説明されています。下のサンプルを参照してください。

H	時間
m	分
s	秒

TimePattern のパターンは、**Time** のパターンに一致する必要があります。出力は**xs:time** 型であるため、出力は常に**YYYY-HH:mm:ss** 構文フォーマットになります。

サンプル

- **altova:parse-time**(xs:string("13:56:24"), "[H]:[m]:[s]") は13:56:24 を返します。
- **altova:parse-time**("13-56-24", "[H]-[m]") は13:56:00 を返します。
- **altova:parse-time**("time=13h56m24s", "time=[H]h[m]m[s]s") は13:56:24 を返します。
- **altova:parse-time**("time=24s56m13h", "time=[s]s[m]m[H]h") は13:56:24 を返します。

[トップ 1492]

年齢に関連した関数 XP3.1 XQ3.1

関数は計算された年齢(i) 入力引数の日付と現在の日付の期間 (ii) 2 つの入力引数の日付の期間 を返します。**altova:age** 関数は、年齢を年数で返します、**altova:age-details** は年齢を年数、月数、日数からなる3 つの整数のシーケンスで返します。

age [altova:]

altova:age(StartDate *as xs:date*) を **xs:integer** とする XP3.1 XQ3.1

引数として提出された開始日から(システムクロックから取得された)現在の日付 までの日数を数えて、あるオブジェクトの年齢の年数である正数を返します。入力引数が1 年より先大きい場合、または一年の場合、返される値は負の数です。

サンプル

If the current date is **2014-01-15**:

- **altova:age**(xs:date("2013-01-15")) は1 を返します。
- **altova:age**(xs:date("2013-01-16")) は0 を返します。
- **altova:age**(xs:date("2015-01-15")) は-1 を返します。
- **altova:age**(xs:date("2015-01-14")) は0 を返します。

altova:age(StartDate *as xs:date*, **EndDate** *as xs:date*) を **xs:integer** とする XP3.1 XQ3.1

最初の引数として提出された開始日から第 2 の引数の終了日 までの日数を数えて、あるオブジェクトの年齢の年数である正数を返します。最初の引数が1 年または第 2 の引数より先後の日付の場合、返される値は負の数です。

サンプル

If the current date is **2014-01-15**:

- **altova:age**(xs:date("2000-01-15"), xs:date("2010-01-15")) は10 を返します。
- **altova:age**(xs:date("2000-01-15"), current-date()) は、現在の日付が2014-01-15 の場合 14 を返します。
- **altova:age**(xs:date("2014-01-15"), xs:date("2010-01-15")) は-4 を返します。

age-details [altova:]

altova:age-details(InputDate *as xs:date*) を(**xs:integer**)* とする XP3.1 XQ3.1

引数とシステムクロックから取得された)現在の日付として提出された日付の間の年数、月数、日数の3 つの整数を返します。 返された years+months+days の合計は、2 つの日付 (入力の日付と現在の日付) の時間差です。 入力の日付は現在の日付より先早いまたはお遅い値をもつことができますが、入力の日付が早いか遅いかは返される値で示されていません。戻される値は常に正の数です。

サンプル

現在の日付が**2014-01-15** の場合:

- **altova:age-details**(xs:date("2014-01-16")) は(0 0 1)を返します。
- **altova:age-details**(xs:date("2014-01-14")) は(0 0 1) を返します。
- **altova:age-details**(xs:date("2013-01-16")) は(1 0 1) を返します。
- **altova:age-details**(current-date()) は(0 0 0)を返します。

altova:age-details(Date-1 *as xs:date*, **Date-2** *as xs:date*) を(**xs:integer**)* とする XP3.1 XQ3.1

二つの引数間の年数、月数、日数の3 つの整数を返します。 返された years+months+days の合計は、2 つの入力の日付の時間差です。最初の引数として提出される二つの日付はどちらが速くても、または遅くてもかまいません。返される値は入力の日付が現在の日付より早いかまたは遅いかを示しません。戻される値は常に正の数です。

サンプル

- **altova:age-details**(xs:date("2014-01-16"), xs:date("2014-01-15")) は (0 0 1) を返します。
- **altova:age-details**(xs:date("2014-01-15"), xs:date("2014-01-16")) は (0 0 1) を返します。

[トップ 1492]

26.2.1.3 XPath/XQuery 関数: 位置情報

以下の位置情報 XPath/XQuery 拡張関数は、XMLSpy の現在のバージョンによりサポートされています。また、次で使用することができます: (i) XSLT コンテキスト内の XPath 式、または (ii) XQuery ドキュメント内の XQuery 式。

関数の名前指定と言語の適用性に関するメモ

Altova 拡張関数は XPath/XQuery 式で使用することができ、XPath、XQuery、および XSLT 関数の標準ライブラリで使用可能な機能に更なる機能性を与えます。Altova 拡張関数は **Altova 拡張関数名前空間**、**http://www.altova.com/xslt-extensions** に収められており、**altova:** プレフィックスが、このセクションでは使用されます。製品の今後のバージョンが拡張機能への継続的サポート、または個別の関数の振る舞いは変更する可能性があることに注意してください。Altova 拡張機能へのサポートに関しては、今後のリリースのドキュメントを参照してください。

XPath 関数 (XSLT 内の XPath 式で使用):	XP1 XP2 XP3.1
XSLT 関数 (XSLT 内の XPath 式で使用):	XSLT1 XSLT2 XSLT3
XQuery 関数 (XQuery 内の XQuery 式で使用):	XQ1 XQ3.1

format-geolocation [altova:]

altova:format-geolocation(Latitude *as xs:decimal*, **Longitude** *as xs:decimal*, **GeolocationOutputStringFormat** *as xs:integer*) を **xs:string** とする XP3.1 XQ3.1

緯度と経度を最初の 2 つの引数とし、位置情報を文字列として出力します。第 3 の引数 **GeolocationOutputStringFormat** は、位置情報出力文字列のフォーマットです。出力文字列のフォーマットを識別するために、正数の値 1 から 4 を使用します。(*下の「位置情報出力文字列フォーマット」を参照してください*)。緯度の値の範囲は +90 から -90 (北 から 南) です。経度の値の範囲は +180 から -180 (東 から 西) です。

メモ image-exif-data 1517 関数と Exif メタデータの @Geolocation 1517 属性を位置情報入力文字列を提供する際に使用することができます。

サンプル

- **altova:format-geolocation**(33.33, -22.22, 4) は xs:string "33.33 -22.22" を返します。
- **altova:format-geolocation**(33.33, -22.22, 2) は xs:string "33.33N 22.22W" を返します。
- **altova:format-geolocation**(-33.33, 22.22, 2) は xs:string "33.33S 22.22E" を返します。
- **altova:format-geolocation**(33.33, -22.22, 1) は xs:string "33°19'48.00"S 22° 13'12.00"E" を返します。

位置情報出力文字列フォーマット:

与えられた緯度と経度は下にリストされる出力フォーマットによりフォーマットされます。希望するフォーマットは整数 ID (1 から 4) により識別されます。緯度の値の範囲は +90 から -90 (北 から 南) です。経度の値の範囲は +180 から -180 (東 から 西) です。

1
度、分、10 進の秒、方角のサフィックス付き (N/S, E/W) **D°M'S.SS"N/S D°M'S.SS"E/W** *サンプル:* **33°55'11.11"N 22°44'66.66"W**

2
10 進の度、方角のサフィックス付き (N/S, E/W) **D.DDN/S D.DDE/W** *サンプル:* **33.33N 22.22W**

3
度、分、10 進の秒、プレフィックスサイン付き (+/-); (**N/E**) のためのプラスサインは任意です。 **+/-D°M'S.SS" +/-D°M'S.SS"** *サンプル:* **33°55'11.11" -22°44'66.66"**

4
10 進の度、プレフィックスサイン付き (+/-); (**N/E**) のためのプラスサインは任意です。 **+/-D.DD +/-D.DD** *サンプル:* **33.33 -22.22**

Altova Exif 属性: 位置情報

Altova XPath/XQuery エンジンはカスタム属性 **Geolocation**を標準 Exif メタデータ タグから生成します。**Geolocation** は、4 つの Exif タグの連結です: 単位の追加された (*下のテーブル参照*) GPSLatitude、GPSLatitudeRef、GPSLongitude、GPSLongitudeRef 。

GPSLatitude	GPSLatitudeRef	GPSLongitude	GPSLongitudeRef	Geolocation
33 51 21.91	S	151 13 11.73	E	33°51'21.91"S 151° 13'11.73"E

parse-geolocation [altova:]

altova:parse-geolocation(GeolocationInputString *as xs:string*) を **xs:decimal+** とする **XP3.1 XQ3.1**

GeolocationInputString 引数を解析して、位置情報の緯度と経度 (この通りの順番) を 2 つの xs:decimal アイテムのシーケンスとして返します。位置情報入力文字列が提供されることのできるフォーマットは以下のリストの通りです。

(C) 2015-2021 Altova GmbH

メモ image-exif-data 1517 関数とExif メタデータの @Geolocation 1517 属性を位置情報入力文字列を提供する際に使用することができます。

サンプル

- **altova:parse-geolocation**("33.33 -22.22") は2 つの xs:decimals (33.33, 22.22) のシーケンスを返します。
- **altova:parse-geolocation**("48°51'29.6""N 24°17'40.2""") は2 つの xs:decimals (48.8582222222222, 24.2945) のシーケンスを返します。
- **altova:parse-geolocation**('48°51''29.6"N 24°17''40.2"') は2 つの xs:decimals (48.8582222222222, 24.2945) のシーケンスを返します。
- **altova:parse-geolocation**(**image-exif-data**(//MyImages/Image20141130.01)**/@Geolocation**) は2 つの xs:decimals のシーケンスを返します。

位置情報入力文字列フォーマット:

位置情報入力文字列は空白で区別された緯度と経度（この通りの順番）を含む必要があります。緯度と経度は以下のフォーマットをとることができます。組み合わせることも可能です。緯度が1つのフォーマットで、経度が他のフォーマットをとることができます。緯度の値の範囲は +90 から -90 (N から S)。経度の値の範囲は +180 から -180 (E から W)。

メモ 単一およびダブル引用符が入力文字列引数を区切るために使用されていると、使用されている単一およびダブル引用が、それぞれ、分の値と秒の値、不一致をもたらします。この様な場合、分の値と秒の値を表すための使用されている引用符は、ダブルとしてエスケープされる必要があります。このセクションのサンプルでは、入力文字列を区別するために使用されている引用符は黄色い（"）でハイライトされており、エスケープした単位インジケーターは青い（""）でハイライトされています。

- 度、分、10 進の秒、方角のサフィックス付き（N/S, E/W）
  D°M'S.SS"N/S D°M'S.SS"W/E
  サンプル: **33°55'11.11"N 22°44'55.25"W**

- 度、分、10 進の秒、プレフィックスサイン付き（+/-）; (**N/E**) のためのプラスサインは任意です。+/-D°M'S.SS" +/-D°M'S.SS"
  サンプル: **33°55'11.11" -22°44'55.25"**

- 度、分、10 進の分、方角のサフィックス付き（N/S, E/W）
  D°M.MM'N/S D°M.MM'W/E
  サンプル: **33°55.55'N 22°44.44'W**

- 度、分、10 進の分、プレフィックスサイン付き（+/-）; (**N/E**) のためのプラスサインは任意です。+/-D°M.MM' +/-D°M.MM'
  サンプル: **+33°55.55' -22°44.44'**

- 10 進の度、方角のサフィックス付き（N/S, E/W）
  D.DDN/S D.DDW/E
  サンプル: **33.33N 22.22W**

- 10 進の度、プレフィックスサイン付き（+/-）; (**N/S E/W**)) のためのプラスサインは任意です。+/-D.DD +/-D.DD
  サンプル: **33.33 -22.22**

フォーマットの組み合わせのサンプル:
33.33N -22°44'55.25"
33.33 22°44'55.25"W

33.33 22.45

Altova Exif 属性: 位置情報

Altova XPath/XQuery エンジンはカスタム属性 Geolocationを標準 Exif メタデータ タグから生成します。Geolocation は、4 つのExif タグの連結です: 単位の追加された(*下のテーブル参照*) GPSLatitude、GPSLatitudeRef、GPSLongitude、GPSLongitudeRef 。

GPSLatitude	GPSLatitudeRef	GPSLongitude	GPSLongitudeRef	Geolocation
33 51 21.91	S	151 13 11.73	E	33°51'21.91"S 151° 13'11.73"E

geolocation-distance-km [altova:]

altova:geolocation-distance-km(GeolocationInputString-1 *as xs:string*, **GeolocationInputString-2** *as xs:string*) を **xs:decimal** とする **XP3.1** **XQ3.1**

2 つの位置情報の間の距離をキロメートルで計算します。位置情報入力文字列を提供することのできるフォーマットは下にリストされています。緯度の値の範囲は +90 から -90 (北 から 南) です。経度の値の範囲は +180 から -180 (東 から 西)です。

メモ image-exif-data 1517 関数とExif メタデータの @Geolocation 1517 属性を位置情報入力文字列を提供する際に使用することができます。

サンプル

- **altova:geolocation-distance-km**("33.33 -22.22", "48°51'29.6""N 24°17'40.2""") は xs:decimal 4183.08132372392 を返します。

位置情報入力文字列フォーマット:

位置情報入力文字列は空白で区別された緯度と経度 (この通りの順番) を含む必要があります。緯度と経度は以下のフォーマットをとることができます。組み合わせることも可能です。緯度が 1 つのフォーマットで、経度が他のフォーマットをとることができます。緯度の値の範囲は +90 から -90 (N から S)。経度の値の範囲は +180 から -180 (E から W)。

メモ 単一およびダブル引用符が入力文字列引数を区切るために使用されていると、使用されている単一およびダブル引用が、それぞれ、分の値と秒の値、不一致をもたらします。この様な場合、分の値と秒の値を表すための使用されている引用符は、ダブルとしてエスケープされる必要があります。このセクションのサンプルでは、入力文字列を区別するために使用されている引用符は黄色い(") でハイライトされており、エスケープした単位インジケーターは青い("") でハイライトされています。

- 度、分、10 進の秒、方角のサフィックス付き (N/S, E/W)
  D°M'S.SS"N/S D°M'S.SS"W/E
  サンプル: **33°55'11.11"N 22°44'55.25"W**

- 度、分、10 進の秒、プレフィックスサイン付き (+/-); (**N/E**) のためのプラスサインは任意です。+/-D°M'S.SS" +/-D°M'S.SS"
  サンプル: **33°55'11.11" -22°44'55.25"**

- 度、分、10 進の分、方角のサフィックス付き (N/S, E/W)
  D°M.MM'N/S D°M.MM'W/E
  サンプル: **33°55.55'N 22°44.44'W**

(C) 2015-2021 Altova GmbH

- 度、分、10 進の分、プレフィックスサイン付き (+/-); (**N/E**) のためのプラスサインは任意です。+/-D°M.MM' +/-D°M.MM'
  サンプル: **+33°55.55' -22°44.44'**

- 10 進の度、方角のサフィックス付き (N/S, E/W)
  D.DDN/S D.DDW/E
  サンプル: **33.33N 22.22W**

- 10 進の度、プレフィックスサイン付き (+/-); (**N/S E/W**)) のためのプラスサインは任意です。+/-D.DD +/-D.DD
  サンプル: **33.33 -22.22**

フォーマットの組み合わせのサンプル:
33.33N -22°44'55.25"
33.33 22°44'55.25"W
33.33 22.45

Altova Exif 属性: 位置情報

Altova XPath/XQuery エンジンはカスタム属性 **Geolocation**を標準 Exif メタデータ タグから生成します。**Geolocation** は、4 つの Exif タグの連結です: 単位の追加された (*下のテーブル参照*) GPSLatitude、GPSLatitudeRef、GPSLongitude、GPSLongitudeRef 。

GPSLatitude	GPSLatitudeRef	GPSLongitude	GPSLongitudeRef	Geolocation
33 51 21.91	S	151 13 11.73	E	33°51'21.91"S 151° 13'11.73"E

geolocation-distance-mi [altova:]

altova:geolocation-distance-mi(GeolocationInputString-1 *as xs:string*, **GeolocationInputString-2** *as xs:string*) を **xs:decimal** とする **XP3.1** **XQ3.1**
2 つの位置情報の間の距離をマイルで計算します。位置情報入力文字列を提供することのできるフォーマットは下にリストされています。緯度の値の範囲は +90 から -90 (北 から 南) です。経度の値の範囲は +180 から -180 (東 から 西)です。

メモ image-exif-data 1517 関数と Exif メタデータの @Geolocation 1517 属性を位置情報入力文字列を提供する際に使用することができます。

サンプル

- **altova:geolocation-distance-mi**("33.33 -22.22", "48°51'29.6""N 24°17'40.2""") は xs:decimal 2599.40652340653 を返します。

位置情報入力文字列フォーマット:

位置情報入力文字列は空白で区別された緯度と経度 (この通りの順番) を含む必要があります。緯度と経度は以下のフォーマットをとることができます。組み合わせることも可能です。緯度が 1 つのフォーマットで、経度が他のフォーマットをとることができます。緯度の値の範囲は +90 から -90 (N から S)。経度の値の範囲は +180 から -180 (E から W)。

メモ 単一およびダブル引用符が入力文字列引数を区切るために使用されていると、使用されている単一およびダブル引用が、それぞれ、分の値と秒の値、不一致をもたらします。この様な場合、分の値と秒の値を表すための使用されている引用符は、ダ

ブルにしてエスケープされる必要があります。このセクションのサンプルでは、入力文字列を区別するために使用されている引用符は黄色い(") でハイライトされており、エスケープした単位インジケーターは青い("") でハイライトされています。

- 度、分、10 進の秒、方角のサフィックス付き(N/S, E/W)
  D°M'S.SS"N/S D°M'S.SS"W/E
  サンプル: **33°55'11.11"N 22°44'55.25"W**

- 度、分、10 進の秒、プレフィックスサイン付き(+/-); (**N/E**) のためのプラスサインは任意です。+/-D°M'S.SS" +/-D°M'S.SS"
  サンプル: **33°55'11.11" -22°44'55.25"**

- 度、分、10 進の分、方角のサフィックス付き (N/S, E/W)
  D°M.MM'N/S D°M.MM'W/E
  サンプル: **33°55.55'N 22°44.44'W**

- 度、分、10 進の分、プレフィックスサイン付き(+/-); (**N/E**) のためのプラスサインは任意です。+/-D°M.MM' +/-D°M.MM'
  サンプル: **+33°55.55' -22°44.44'**

- 10 進の度、方角のサフィックス付き(N/S, E/W)
  D.DDN/S D.DDW/E
  サンプル: **33.33N 22.22W**

- 10 進の度、プレフィックスサイン付き(+/-); (**N/S E/W**)) のためのプラスサインは任意です。+/-D.DD +/-D.DD
  サンプル: **33.33 -22.22**

フォーマットの組み合わせのサンプル:
33.33N -22°44'55.25"
33.33 22°44'55.25"W
33.33 22.45

Altova Exif 属性: 位置情報

Altova XPath/XQuery エンジンはカスタム属性 **Geolocation**を標準 Exif メタデータ タグから生成します。**Geolocation** は、4 つのExif タグの連結です: 単位の追加された(*下のテーブル参照*) GPSLatitude、GPSLatitudeRef、GPSLongitude、GPSLongitudeRef 。

GPSLatitude	GPSLatitudeRef	GPSLongitude	GPSLongitudeRef	Geolocation
33 51 21.91	S	151 13 11.73	E	33°51'21.91"S 151° 13'11.73"E

geolocations-bounding-rectangle [altova:]

altova:geolocations-bounding-rectangle(Geolocations を *xs:sequence* として GeolocationOutputStringFormat を *xs:integer* として) を**xs:string** として **XP3.1** **XQ3.1**
最初の引数として文字列のシーケンスを取ります。シーケンスの各文字列が位置情報です。関数はそれぞれが、最初の引数に送信されたすべての位置情報を含むために最適にサイズ調整された長方形の左上と右下の位置情報の座標である2 つの文字列のシーケンスを返します。位置情報入力文字列が表示される書式は以下にリストされています(*「位置情報入力文字列書式」を参照してください*)。緯度の値は+90 から-90 への(N からS への) の範囲です。経度の値は+180 から-180 への(E からW への) の範

囲です。

関数の2番目の引数は出力シーケンス内の2つの位置情報文字列の書式を指定します。引数はそれぞれの値が異なる位置情報文字列書式を識別する 1 から 4 の整数の値を取ります（*下記の「位置情報出力文字列書式」を参照してください*）。

メモ image-exif-data 1517 関数と Exif メタデータの属性は入力文字列を提供するために使用することができます。

サンプル

- **altova:geolocations-bounding-rectangle**(("48.2143531 16.3707266", "51.50939 -0.11832"), 1) はシーケンス ("51°30'33.804"N 0°7'5.952"W", "48°12'51.67116"N 16°22'14.61576"E") を返します。
- **altova:geolocations-bounding-rectangle**(("48.2143531 16.3707266", "51.50939 -0.11832", "42.5584577 -70.8893334"), 4) はシーケンス ("51.50939 -70.8893334", "42.5584577 16.3707266") を返します

位置情報入力文字列フォーマット:

位置情報入力文字列は空白で区別された緯度と経度（この通りの順番）を含む必要があります。緯度と経度は以下のフォーマットをとることができます。組み合わせることも可能です。緯度が1つのフォーマットで、経度が他のフォーマットをとることができます。緯度の値の範囲は +90 から -90 (N から S)。経度の値の範囲は +180 から -180 (E から W)。

メモ 単一およびダブル引用符が入力文字列引数を区切るために使用されていると、使用されている単一およびダブル引用が、それぞれ、分の値と秒の値、不一致をもたらします。この様な場合、分の値と秒の値を表すための使用されている引用符は、ダブルとしてエスケープされる必要があります。このセクションのサンプルでは、入力文字列を区別するために使用されている引用符は黄色い (") でハイライトされており、エスケープした単位インジケーターは青い ("") でハイライトされています。

- 度、分、10 進の秒、方角のサフィックス付き (N/S, E/W)
  D°M'S.SS"N/S D°M'S.SS"W/E
  サンプル: **33°55'11.11"N 22°44'55.25"W**

- 度、分、10 進の秒、プレフィックスサイン付き (+/-); (**N/E**) のためのプラスサインは任意です。+/-D°M'S.SS" +/-D°M'S.SS"
  サンプル: **33°55'11.11" -22°44'55.25"**

- 度、分、10 進の分、方角のサフィックス付き (N/S, E/W)
  D°M.MM'N/S D°M.MM'W/E
  サンプル: **33°55.55'N 22°44.44'W**

- 度、分、10 進の分、プレフィックスサイン付き (+/-); (**N/E**) のためのプラスサインは任意です。+/-D°M.MM' +/-D°M.MM'
  サンプル: **+33°55.55' -22°44.44'**

- 10 進の度、方角のサフィックス付き (N/S, E/W)
  D.DDN/S D.DDW/E
  サンプル: **33.33N 22.22W**

- 10 進の度、プレフィックスサイン付き (+/-); (**N/S E/W**)) のためのプラスサインは任意です。+/-D.DD +/-D.DD
  サンプル: **33.33 -22.22**

フォーマットの組み合わせのサンプル:
33.33N -22°44'55.25"
33.33 22°44'55.25"W

```
33.33  22.45
```

位置情報出力文字列フォーマット:

与えられた緯度と経度は下にリストされる出力フォーマットによりフォーマットされます。希望するフォーマットは整数 ID (1 から 4) により識別されます。緯度の値の範囲は +90 から -90 (北 から 南) です。経度の値の範囲は +180 から -180 (東 から 西) です。

1
度、分、10 進の秒、方角のサフィックス付き (N/S, E/W) D°M'S.SS"N/S D°M'S.SS"E/W *サンプル:* 33°55'11.11"N 22°44'66.66"W

2
10 進の度、方角のサフィックス付き (N/S, E/W) D.DDN/S D.DDE/W *サンプル:* 33.33N 22.22W

3
度、分、10 進の秒、プレフィックスサイン付き (+/-); (N/E) のためのプラスサインは任意です。 +/-D°M'S.SS" +/-D°M'S.SS" *サンプル:* 33°55'11.11" -22°44'66.66"

4
10 進の度、プレフィックスサイン付き (+/-); (N/E) のためのプラスサインは任意です。 +/-D.DD +/-D.DD *サンプル:* 33.33 -22.22

Altova Exif 属性: 位置情報

Altova XPath/XQuery エンジンはカスタム属性 Geolocationを標準 Exif メタデータ タグから生成します。Geolocation は、4 つのExif タグの連結です: 単位の追加された (*下のテーブル参照*) GPSLatitude、GPSLatitudeRef、GPSLongitude、GPSLongitudeRef 。

GPSLatitude	GPSLatitudeRef	GPSLongitude	GPSLongitudeRef	Geolocation
33 51 21.91	S	151 13 11.73	E	33°51'21.91"S 151° 13'11.73"E

geolocation-within-polygon [altova:]

altova:geolocation-within-polygon(Geolocation *as xs:string*, **((PolygonPoint** *as xs:string*)+)) を **xs:boolean** とする **XP3.1** **XQ3.1**

PolygonPoint 引数により説明されている **Geolocation**（最初の引数）が多角形のエリア内に存在するかを決定します。もし、PolygonPoint 引数が最初と最後のポイントが同じ場合に作成される閉じられたフィギュアを作成しない場合、フィギュアを閉じるために、最初のポイントが明示的に最後のポイントとして追加されます。全ての引数(Geolocation および PolygonPoint+）は、（*下にリストされるフォーマットの*）位置情報入力文字列により提供されます。Geolocation 引数が多角形エリア内にある場合、関数は true(); を返します。その他の場合は false() を返します。緯度の値の範囲は +90 から -90（北 から 南）です。経度の値の範囲は +180 から -180（東 から 西）です。

メモ image-exif-data 1517 関数と Exif メタデータの @Geolocation 1517 属性を位置情報入力文字列を提供する際に使用することができます。

サンプル

- **altova:geolocation-within-polygon("33 -22", ("58 -32", "-78 -55", "48 24", "58 -32"))** は true() を返します。
- **altova:geolocation-within-polygon("33 -22", ("58 -32", "-78 -55", "48 24"))** は true() を返します。
- **altova:geolocation-within-polygon("33 -22", ("58 -32", "-78 -55", "48° 51'29.6""N 24°17'40.2""")))** は true() を返します。

位置情報入力文字列フォーマット:

位置情報入力文字列は空白で区別された緯度と経度（この通りの順番）を含む必要があります。緯度と経度は以下のフォーマットをとることができます。組み合わせることも可能です。緯度が 1 つのフォーマットで、経度が他のフォーマットをとることができます。緯度の値の範囲は +90 から -90 (N から S)。経度の値の範囲は +180 から -180 (E から W)。

メモ 単一およびダブル引用符が入力文字列引数を区切るために使用されていると、使用されている単一およびダブル引用が、それぞれ、分の値と秒の値、不一致をもたらします。この様な場合、分の値と秒の値を表すための使用されている引用符は、ダブルにしてエスケープされる必要があります。このセクションのサンプルでは、入力文字列を区別するために使用されている引用符は黄色い(") でハイライトされており、エスケープした単位インジケーターは青い("") でハイライトされています。

- 度、分、10 進の秒、方角のサフィックス付き（N/S, E/W）
  D°M'S.SS"N/S D°M'S.SS"W/E
  サンプル: **33°55'11.11"N 22°44'55.25"W**

- 度、分、10 進の秒、プレフィックスサイン付き（+/-）; (**N/E**) のためのプラスサインは任意です。+/-D°M'S.SS" +/-D°M'S.SS"
  サンプル: **33°55'11.11" -22°44'55.25"**

- 度、分、10 進の分、方角のサフィックス付き（N/S, E/W）
  D°M.MM'N/S D°M.MM'W/E
  サンプル: **33°55.55'N 22°44.44'W**

- 度、分、10 進の分、プレフィックスサイン付き（+/-）; (**N/E**) のためのプラスサインは任意です。+/-D°M.MM' +/-D°M.MM'
  サンプル: **+33°55.55' -22°44.44'**

- 10 進の度、方角のサフィックス付き（N/S, E/W）
  D.DDN/S D.DDW/E
  サンプル: **33.33N 22.22W**

- 10 進の度、プレフィックスサイン付き（+/-）; (**N/S E/W**)) のためのプラスサインは任意です。+/-D.DD +/-D.DD
  サンプル: **33.33 -22.22**

フォーマットの組み合わせのサンプル:

```
33.33N  -22°44'55.25"
33.33  22°44'55.25"W
33.33  22.45
```

Altova Exif 属性: 位置情報

Altova XPath/XQuery エンジンはカスタム属性 **Geolocation**を標準 Exif メタデータ タグから生成します。**Geolocation** は、4 つのExif タグの連結です: 単位の追加された(*下のテーブル参照*) GPSLatitude、GPSLatitudeRef、GPSLongitude、GPSLongitudeRef 。

GPSLatitude	GPSLatitudeRef	GPSLongitude	GPSLongitudeRef	Geolocation
33 51 21.91	S	151 13 11.73	E	33°51'21.91"S 151° 13'11.73"E

geolocation-within-rectangle [altova:]

altova:geolocation-within-rectangle(Geolocation *as xs:string*, **RectCorner-1** *as xs:string*, **RectCorner-2** *as xs:string*) を**xs:boolean** とする **XP3.1** **XQ3.1**

長方形の対格の角を指定する第 2 及び第 3 引数(**RectCorner-1**、**RectCorner-2**)により説明されている **Geolocation**(最初の引数) が長方形のエリア内に存在するかを決定します。全ての引数(Geolocation、**RectCorner-1** および**RectCorner-2**) は、(*下にリストされるフォーマットの*) 位置情報入力文字列により提供されます。Geolocation 引数が長方形エリア内にある場合、関数は true(); を返します。その他の場合は false() を返します。緯度の値の範囲は +90 から -90 (北 から 南) です。経度の値の範囲は +180 から -180 (東 から 西)です。

メモ image-exif-data 1517 関数とExif メタデータの @Geolocation 1517 属性を位置情報入力文字列を提供する際に使用することができます。

サンプル

- **altova:geolocation-within-rectangle("**33 -22", "58 -32", "-48 24") は true() を返します。
- **altova:geolocation-within-rectangle("**33 -22", "58 -32", "48 24") は false() を返します。
- **altova:geolocation-within-rectangle("**33 -22", "58 -32", "48°51'29.6""S 24° 17'40.2""") は true() を返します。

位置情報入力文字列フォーマット:

位置情報入力文字列は空白で区別された緯度と経度 (この通りの順番) を含む必要があります。緯度と経度は以下のフォーマットをとることができます。組み合わせることも可能です。緯度が 1 つのフォーマットで、経度が他のフォーマットをとることができます。緯度の値の範囲は +90 から -90 (N から S)。経度の値の範囲は +180 から -180 (E から W)。

メモ 単一およびダブル引用符が入力文字列引数を区切るために使用されていると、使用されている単一およびダブル引用が、それぞれ、分の値と秒の値、不一致をもたらします。この様な場合、分の値と秒の値を表すための使用されている引用符は、ダブルとしてエスケープされる必要があります。このセクションのサンプルでは、入力文字列を区別するために使用されている引用符は黄色い(") でハイライトされており、エスケープした単位インジケーターは青い("") でハイライトされています。

- 度、分、10 進の秒、方角のサフィックス付き(N/S, E/W)

D°M'S.SS"N/S D°M'S.SS"W/E
サンプル: 33°55'11.11"N 22°44'55.25"W

- 度、分、10 進の秒、プレフィックスサイン付き(+/-); (N/E) のためのプラスサインは任意です。+/-D°M'S.SS" +/-D°M'S.SS"
  サンプル: 33°55'11.11" -22°44'55.25"

- 度、分、10 進の分、方角のサフィックス付き (N/S, E/W)
  D°M.MM'N/S D°M.MM'W/E
  サンプル: 33°55.55'N 22°44.44'W

- 度、分、10 進の分、プレフィックスサイン付き(+/-); (N/E) のためのプラスサインは任意です。+/-D°M.MM' +/-D°M.MM'
  サンプル: +33°55.55' -22°44.44'

- 10 進の度、方角のサフィックス付き(N/S, E/W)
  D.DDN/S D.DDW/E
  サンプル: 33.33N 22.22W

- 10 進の度、プレフィックスサイン付き(+/-); (N/S E/W)) のためのプラスサインは任意です。+/-D.DD +/-D.DD
  サンプル: 33.33 -22.22

フォーマットの組み合わせのサンプル:
33.33N -22°44'55.25"
33.33 22°44'55.25"W
33.33 22.45

Altova Exif 属性: 位置情報

Altova XPath/XQuery エンジンはカスタム属性 Geolocationを標準 Exif メタデータ タグから生成します。Geolocation は、4 つのExif タグの連結です: 単位の追加された(*下のテーブル参照*) GPSLatitude、GPSLatitudeRef、GPSLongitude、GPSLongitudeRef 。

GPSLatitude	GPSLatitudeRef	GPSLongitude	GPSLongitudeRef	Geolocation
33 51 21.91	S	151 13 11.73	E	33°51'21.91"S 151° 13'11.73"E

[トップ 1507]

26.2.1.4 XPath/XQuery 関数: イメージに関連

以下のイメージに関連したXPath/XQuery 拡張関数は、XMLSpy の現在のバージョンによりサポートされています。また、次で使用することができます: (i) XSLT コンテキスト内のXPath 式、または(ii) XQuery ドキュメント内のXQuery 式。

関数の名前指定と言語の適用性に関するメモ

Altova 拡張関数はXPath/XQuery 式で使用することができ、XPath、XQuery、およびXSLT 関数の標準ライブラリで使用可能

な機能に更なる機能性を与えます。Altova 拡張関数はAltova **拡張関数名前空間**、**http://www.altova.com/xslt-extensions** に収められており、**altova:** プレフィックスが、このセクションでは使用されます。製品の今後のバージョンが拡張機能への継続的サポート、または個別の関数の振る舞いは変更する可能性があることに注意してください。Altova 拡張機能へのサポートに関しては、今後のリリースのドキュメントを参照してください。

XPath 関数 (XSLT 内の XPath 式で使用):	XP1 XP2 XP3.1
XSLT 関数 (XSLT 内の XPath 式で使用):	XSLT1 XSLT2 XSLT3
XQuery 関数 (XQuery 内の XQuery 式で使用):	XQ1 XQ3.1

suggested-image-file-extension [altova:]

altova:suggested-image-file-extension(Base64String *as string*) を **string?** とする XP3.1 XQ3.1

イメージファイルのBase64 エンコード を引数として、イメージのファイル拡張子を、イメージのBase64エンコード内の記録として返します。返された値は、エンコード内で使用することのできるイメージ型情報を基にしたヒントです。この情報が使用できない場合、空の文字列が返されます。この関数は、Base64 イメージをファイルとして保存し、適切なファイル拡張子を動的取得する際に役に立ちます。

サンプル

- **altova:suggested-image-file-extension**(/MyImages/MobilePhone/Image20141130.01) は 'jpg' を返します。
- **altova:suggested-image-file-extension**($XML1/Staff/Person/@photo) は '' を返します。

上のサンプルでは、関数の引数として与えられたノードはBase64 エンコードイメージを含むと仮定します。最初のサンプルは jpg をファイルの型および拡張子として取得します。二番目のサンプルでは、与えられたBase64 エンコードは使用できる拡張子の情報を提供しません。

mt-transform-image [altova:]

altova:mt-transform-image(Base64Image *as Base64BinaryString*, **Size** *as item()+*, **Rotation** *as xs:integer*, **Quality** *as xs:integer*) を **Base64BinaryString** とする XP3.1 XQ3.1

Base64-エンコード イメージ を最初の引数として、変換されたBase64-エンコード イメージを返します。第 2、第 3、第 4 引数は変換された以下のイメージ パラメーターです: サイズ、回転、およびクオリティ。

- サイズ引数には3 つのサイズ変更のオプションがあります。

(X,Y)	絶対ピクセルの値。アスペクト率は保持されません。高さと幅は自動的にイメージの長いおよび短い辺に合わされるため、高さと幅の指示は関係ありません。2 つの整数アイテムのシーケンスとして値が入力されます。かっこが必要です。
X	X をピクセルの新しい長い辺として、イメージの縦横比のサイズ変更が行われます。アスペクト率は保持されます。値は整数で引用符なしで入力されます。
'X%'	元のディメンションの与えられたパーセンテージにイメージのサイズを変更します。値は文字列として引用符を付けて入力される必要があります。

- 回転は以下の値を持つことができます: 90、180、270、-90、-180、-270。これらの値は回転の度数です。正の値はイメージを時計回りに回転させます。負の値はイメージを反時計回りに回転させます。 Altova Exif 属性 **OrientationDegree** を使用して、イメージの現在の回転の度数(0, 90, 180, 270) をイメージのExif **Orientation** タグから取得することができます。ですが、**OrientationDegree** 属性はデータの**Orientation** タグから取得されるため、Exif データ内に**Orientation** タグが存在する場合のみ使用することができます。(*下の*

OrientationDegree 説明を参照してください)。

- クオリティは、0 から100 の値で、JPEG 圧縮のIJG クオリティスケールの値を参照していますが、クオリティのパーセンテージのインジケーターではありません。サイズとクオリティのどちらかを優先すると、もう一方の優先度が下がります。フルカラーソースのイメージでは、75 が通常最高値と見なされています。もし、75 が満足のいく結果をもたらさない場合は、値を上げてください。

メモ Exif データが元のイメージに存在する場合、変換時に削除され、変換されたイメージにはExif データは存在しません。

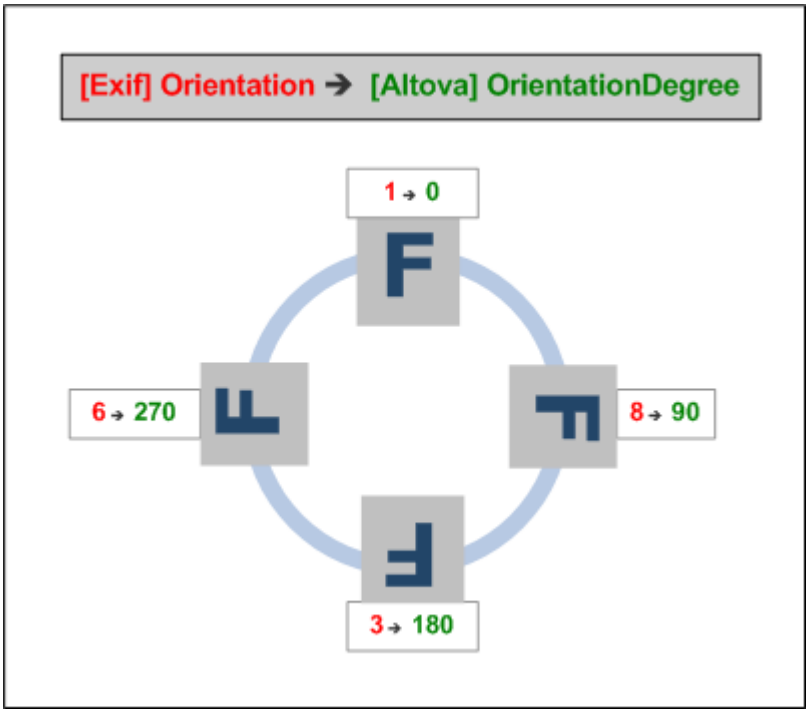
サンプル

- **mt-transform-image**(Images/Image[@id='43'], '50%', 90, 75)
  関数は、43 の@id 値を持つイメージ子孫ノード内でBase64-エンコード 文字列として保管されているイメージを入力とします。関数は変換されたイメージを返します。変換されたイメージは50% まで、サイズ変更され、時計回りに90 度回転され、75 のクオリティレベルを与えられます。
- **mt-transform-image**(Images/Image[@id='43'], 400, 90, 75)
  関数は前のサンプルと同じ結果を出しますが、長い辺は400 ピクセルの特定の値に設定されています。元のイメージのアスペクト率は保持されます。
- **mt-transform-image**(Images/Image[@id='43'], (400, 280), image-exif-data($XML1/$XML1/Images/ReferenceImage)/@OrientationDegree, 75)
  このサンプルは、前のサンプルと同じイメージを選択し、同じクオリティ値(75) を設定します。イメージのサイズは400x280 ピクセルに設定されています。回転値は、ノード内のBase64-エンコード イメージの**@OrientationDegree** 属性から得られます。

Altova Exif 属性: OrientationDegree

Altova XPath/XQuery エンジンはカスタム属性 **OrientationDegree** をExif メタデータタグ **Orientation** から生成します。

OrientationDegree は標準Exif タグ Orientation を正数の値(1、8、3 または6) からそれぞれ対応する度数の値(0, 90, 180, 270) へ下の図に示されているように変換します。2、4、5、7 のOrientation 値の変換はないことに注意してください。(これらの向きはイメージ1 を垂直方向中央軸で反転して、2 の値を持つイメージを取得します。そして、このイメージを90-度ごとに時計回りにジャンプさせそれぞれ7、4、および5 の値を取得します。)。



標準 Exif メタタグ

- ImageWidth
- ImageLength
- BitsPerSample
- Compression
- PhotometricInterpretation
- Orientation
- SamplesPerPixel
- PlanarConfiguration
- YCbCrSubSampling
- YCbCrPositioning
- XResolution
- YResolution
- ResolutionUnit
- StripOffsets
- RowsPerStrip
- StripByteCounts
- JPEGInterchangeFormat
- JPEGInterchangeFormatLength
- TransferFunction
- WhitePoint
- PrimaryChromaticities
- YCbCrCoefficients
- ReferenceBlackWhite
- DateTime
- ImageDescription

(C) 2015-2021 Altova GmbH

- Make
- Model
- Software
- Artist
- Copyright

- ExifVersion
- FlashpixVersion
- ColorSpace
- ComponentsConfiguration
- CompressedBitsPerPixel
- PixelXDimension
- PixelYDimension
- MakerNote
- UserComment
- RelatedSoundFile
- DateTimeOriginal
- DateTimeDigitized
- SubSecTime
- SubSecTimeOriginal
- SubSecTimeDigitized
- ExposureTime
- FNumber
- ExposureProgram
- SpectralSensitivity
- ISOSpeedRatings
- OECF
- ShutterSpeedValue
- ApertureValue
- BrightnessValue
- ExposureBiasValue
- MaxApertureValue
- SubjectDistance
- MeteringMode
- LightSource
- Flash
- FocalLength
- SubjectArea
- FlashEnergy
- SpatialFrequencyResponse
- FocalPlaneXResolution
- FocalPlaneYResolution
- FocalPlaneResolutionUnit
- SubjectLocation
- ExposureIndex
- SensingMethod
- FileSource
- SceneType
- CFAPattern
- CustomRendered
- ExposureMode
- WhiteBalance
- DigitalZoomRatio

- FocalLengthIn35mmFilm
- SceneCaptureType
- GainControl
- Contrast
- Saturation
- Sharpness
- DeviceSettingDescription
- SubjectDistanceRange
- ImageUniqueID

- GPSVersionID
- GPSLatitudeRef
- GPSLatitude
- GPSLongitudeRef
- GPSLongitude
- GPSAltitudeRef
- GPSAltitude
- GPSTimeStamp
- GPSSatellites
- GPSStatus
- GPSMeasureMode
- GPSDOP
- GPSSpeedRef
- GPSSpeed
- GPSTrackRef
- GPSTrack
- GPSImgDirectionRef
- GPSImgDirection
- GPSMapDatum
- GPSDestLatitudeRef
- GPSDestLatitude
- GPSDestLongitudeRef
- GPSDestLongitude
- GPSDestBearingRef
- GPSDestBearing
- GPSDestDistanceRef
- GPSDestDistance
- GPSProcessingMethod
- GPSAreaInformation
- GPSDateStamp
- GPSDifferential

image-exif-data [altova:]

altova:image-exif-data(Base64BinaryString *as string*) を **element?** とする **XP3.1** **XQ3.1**

Base64 エンコード JPEG イメージを引数として、イメージの Exif メタデータを含む **Exif** という名の要素を返します。Exif メタデータは `Exif` 要素の属性の値ペアとして作成されます。属性名は、Base64 エンコード内で検出された Exif データタグです。Exif 仕様タグのリストは以下の通りです。ベンダー特有のタグが Exif データ内に存在する場合、タグとその値も属性値のペアとして返されます。標準 Exif メタデータタグに追加して（*下のリスト参照*）Altova 特有の属性値のペアもまた生成されます。これらの Altova Exif 属性は以下のとおりです。

(C) 2015-2021 Altova GmbH

サンプル

- 属性にアクセスするには、以下の関数を使用します:
  `image-exif-data(//MyImages/Image20141130.01)/@GPSLatitude`
  `image-exif-data(//MyImages/Image20141130.01)/@Geolocation`
- 全ての属性にアクセスするには、以下の関数を使用します:
  `image-exif-data(//MyImages/Image20141130.01)/@*`
- 全ての属性の名前にアクセスするには、以下の式を使用します:
  `for $i in image-exif-data(//MyImages/Image20141130.01)/@* return name($i)`
  関数により返される属性の名前を検出するために役に立ちます。

Altova Exif 属性: 位置情報

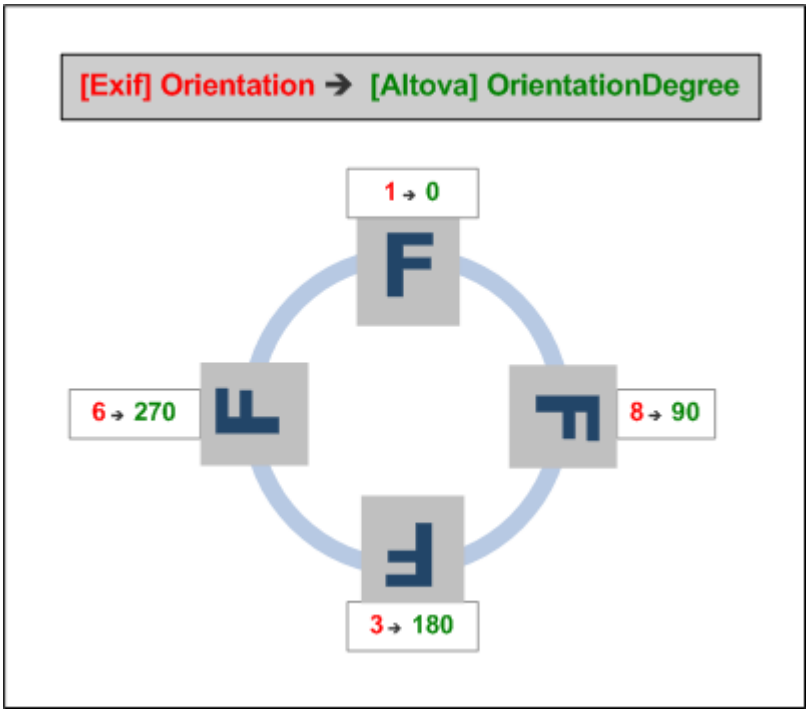
Altova XPath/XQuery エンジンはカスタム属性 **Geolocation**を標準 Exif メタデータ タグから生成します。**Geolocation** は、4 つの Exif タグの連結です: 単位の追加された (下のテーブル参照) GPSLatitude、GPSLatitudeRef、GPSLongitude、GPSLongitudeRef 。

GPSLatitude	GPSLatitudeRef	GPSLongitude	GPSLongitudeRef	Geolocation
33 51 21.91	S	151 13 11.73	E	33°51'21.91"S 151° 13'11.73"E

Altova Exif 属性: OrientationDegree

Altova XPath/XQuery エンジンはカスタム属性 **OrientationDegree** を Exif メタデータタグ **Orientation** から生成します。

OrientationDegree は標準 Exif タグ Orientation を正数の値 (1、8、3 または 6) からそれぞれ対応する度数の値 (0, 90, 180, 270) へ下の図に示されているように変換します。2、4、5、7 の Orientation 値の変換はないことに注意してください。(これらの向きはイメージ 1 を垂直方向中央軸で反転して、2 の値を持つイメージを取得します。そして、このイメージを 90-度ごとに時計回りにジャンプさせそれぞれ 7、4、および 5 の値を取得します。)。



標準 Exif メタデータ タグのリスト

- ImageWidth
- ImageLength
- BitsPerSample
- Compression
- PhotometricInterpretation
- Orientation
- SamplesPerPixel
- PlanarConfiguration
- YCbCrSubSampling
- YCbCrPositioning
- XResolution
- YResolution
- ResolutionUnit
- StripOffsets
- RowsPerStrip
- StripByteCounts
- JPEGInterchangeFormat
- JPEGInterchangeFormatLength
- TransferFunction
- WhitePoint
- PrimaryChromaticities
- YCbCrCoefficients
- ReferenceBlackWhite
- DateTime
- ImageDescription

(C) 2015-2021 Altova GmbH

- Make
- Model
- Software
- Artist
- Copyright

- ExifVersion
- FlashpixVersion
- ColorSpace
- ComponentsConfiguration
- CompressedBitsPerPixel
- PixelXDimension
- PixelYDimension
- MakerNote
- UserComment
- RelatedSoundFile
- DateTimeOriginal
- DateTimeDigitized
- SubSecTime
- SubSecTimeOriginal
- SubSecTimeDigitized
- ExposureTime
- FNumber
- ExposureProgram
- SpectralSensitivity
- ISOSpeedRatings
- OECF
- ShutterSpeedValue
- ApertureValue
- BrightnessValue
- ExposureBiasValue
- MaxApertureValue
- SubjectDistance
- MeteringMode
- LightSource
- Flash
- FocalLength
- SubjectArea
- FlashEnergy
- SpatialFrequencyResponse
- FocalPlaneXResolution
- FocalPlaneYResolution
- FocalPlaneResolutionUnit
- SubjectLocation
- ExposureIndex
- SensingMethod
- FileSource
- SceneType
- CFAPattern
- CustomRendered
- ExposureMode
- WhiteBalance
- DigitalZoomRatio

- FocalLengthIn35mmFilm
- SceneCaptureType
- GainControl
- Contrast
- Saturation
- Sharpness
- DeviceSettingDescription
- SubjectDistanceRange
- ImageUniqueID

- GPSVersionID
- GPSLatitudeRef
- GPSLatitude
- GPSLongitudeRef
- GPSLongitude
- GPSAltitudeRef
- GPSAltitude
- GPSTimeStamp
- GPSSatellites
- GPSStatus
- GPSMeasureMode
- GPSDOP
- GPSSpeedRef
- GPSSpeed
- GPSTrackRef
- GPSTrack
- GPSImgDirectionRef
- GPSImgDirection
- GPSMapDatum
- GPSDestLatitudeRef
- GPSDestLatitude
- GPSDestLongitudeRef
- GPSDestLongitude
- GPSDestBearingRef
- GPSDestBearing
- GPSDestDistanceRef
- GPSDestDistance
- GPSProcessingMethod
- GPSAreaInformation
- GPSDateStamp
- GPSDifferential

[トップ 1517]

26.2.1.5 XPath/XQuery 関数: 数値

Altova の数値拡張関数は XPath と XQuery 内で使用することができ、データを更に処理するための追加機能を提供します。このセクションの関数は Altova の **XPath 3.0** と **XQuery 3.0** エンジンと使用することができます。XPath/XQuery コンテキストで使用することができます。

関数の名前指定と言語の適用性に関するメモ

Altova 拡張関数はXPath/XQuery 式で使用することができ、XPath、XQuery、およびXSLT 関数の標準ライブラリで使用可能な機能に更なる機能性を与えます。Altova 拡張関数はAltova **拡張関数名前空間**、**http://www.altova.com/xslt-extensions** に収められており、**altova:** プレフィックスが、このセクションでは使用されます。製品の今後のバージョンが拡張機能への継続的サポート、または個別の関数の振る舞いは変更する可能性があることご注意してください。Altova 拡張機能へのサポートに関しては、今後のリリースのドキュメントを参照してください。

XPath 関数 (XSLT 内のXPath 式で使用):	XP1 XP2 XP3.1
XSLT 関数 (XSLT 内のXPath 式で使用):	XSLT1 XSLT2 XSLT3
XQuery 関数 (XQuery 内のXQuery 式で使用):	XQ1 XQ3.1

自動付番関数

generate-auto-number [altova:]

altova:generate-auto-number(ID *as xs:string*, **StartsWith** *as xs:double*, **Increment** *as xs:double*, **ResetOnChange** *as xs:string*) を xs:integer とする XP1 XP2 XQ1 XP3.1 XQ3.1

関数が呼び出される度に番号を生成します。関数が初めて呼び出される際に生成される最初の番号は StartsWith 引数により指定されます。その後の関数の呼び出しは新しい番号を生成します。この番号は前に生成された番号を、Increment 引数で指定された値ごとにインクリメントします。実質的には、altova:generate-auto-number 関数は、ID 引数により指定されたカウンターを作成し、このカウンターが関数が呼び出されるごとにインクリメントされます。ResetOnChange 引数の値が、前の関数呼び出しから変更されると、生成される番号の値が StartsWith 値にリセットされます。自動付番は altova:reset-auto-number[1559] 関数を使用してリセットすることができます。

サンプル

- **altova:generate-auto-number**("ChapterNumber", 1, 1, "SomeString") は、関数が呼び出される度に番号を 1 つ返します。1 から始まり、関数が呼び出される度に 1 ずつインクリメントします。その後の呼び出しの 4 番目の"SomeString"引数がである限り、インクリメントは継続されます。4 番目の引数の値が変更されると、(ChapterNumber と呼ばれる) カウンターは 1 にリセットされます。ChapterNumber の値も altova:reset-auto-number[1559] 関数の呼び出しによりの altova:reset-auto-number[1559] ("ChapterNumber") ようにリセットされます。

reset-auto-number [altova:]

altova:reset-auto-number(ID *as xs:string*) XP1 XP2 XQ1 XP3.1 XQ3.1

この関数は、ID 引数内で名づけられた自動付番カウンターの番号をリセットします。引数内のカウンター名を作成した関数の引数により指定された数値にリセットされます。altova:generate-auto-number[1559]

サンプル

- **altova:reset-auto-number**("ChapterNumber") は、altova:generate-auto-number[1559] 関数により作成された ChapterNumber という名の自動付番カウンターをリセットします。
- ChapterNumber を作成した altova:generate-auto-number[1559] 関数の StartsWith 引数の値に数値をリセットします。

[[トップ][1526]]

数値関数

hex-string-to-integer [altova:]

altova:hex-string-to-integer(HexString *as xs:string*) を **xs:integer** とする **XP3.1** **XQ3.1**
10 進のシステム(10進)内の正数の16 進の同値の文字列引数を必要とし、10 進の整数を返します。

サンプル

- **altova:hex-string-to-integer**('1') は1 を返します。
- **altova:hex-string-to-integer**('9') は9 を返します。
- **altova:hex-string-to-integer**('A') は10 を返します。
- **altova:hex-string-to-integer**('B') は11 を返します。
- **altova:hex-string-to-integer**('F') は15 を返します。
- **altova:hex-string-to-integer**('G') はエラーを返します。
- **altova:hex-string-to-integer**('10') は16 を返します。
- **altova:hex-string-to-integer**('01') は1 を返します。
- **altova:hex-string-to-integer**('20') は32 を返します。
- **altova:hex-string-to-integer**('21') は33 を返します。
- **altova:hex-string-to-integer**('5A') は90 を返します。
- **altova:hex-string-to-integer**('USA') はエラーを返します。

integer-to-hex-string [altova:]

altova:integer-to-hex-string(Integer *as xs:integer*) を **xs:string** とする **XP3.1** **XQ3.1**
正数を引数として必要とし、文字列として自身のベース16 の同値を返します。

サンプル

- **altova:integer-to-hex-string**(1) は'1' を返します。
- **altova:integer-to-hex-string**(9) は'9' を返します。
- **altova:integer-to-hex-string**(10) は'A' を返します。
- **altova:integer-to-hex-string**(11) は'B' を返します。
- **altova:integer-to-hex-string**(15) は'F' を返します。
- **altova:integer-to-hex-string**(16) は'10' を返します。
- **altova:integer-to-hex-string**(32) は'20' を返します。
- **altova:integer-to-hex-string**(33) は'21' を返します。
- **altova:integer-to-hex-string**(90) は'5A' を返します。

[[トップ] 1526]

数値フォーマット関数

[[トップ] 1526]

26.2.1.6 XPath/XQuery 関数: スキーマ

下にリストされる Altova 拡張関数はスキーマの情報を返します。以下は関数の詳細 (i) サンプル(ii) スキーマコンポーネントのリストとパースペクティブのプロパティです。 Altova の**XPath 3.0** と**XQuery 3.0** エンジンと共に使用することができ XPath/XQuery コンテキスト内で見つけることができます。

スキーマドキュメントからのスキーマ情報
関数 **altova:schema** には以下の2つの引数が存在します: 1つのゼロ引数と他の2つの引数。ゼロ引数関数はスキーマ全体を返します。この開始点からスキーマ内をナビゲートし必要とするスキーマコンポーネントをロケートします。2-引数関数は自身のQNameにより識別される特定のコンポーネントの型を返します。両方の場合、戻り値は関数です。返されたコンポーネント内をナビゲートするには、特定のコンポーネントのプロパティを選択する必要があります。プロパティが非動的アイテム(すなわちコンポーネントの場合) このコンポーネントのプロパティをさらに選択してナビゲートすることが可能です。選択されたプロパティが動的なアイテムの場合、アイテムの値が返され、ナビゲートを更に行うことはできません。

メモ Xquery 式ではスキーマは明示的にインポートされる必要があります。XPath 式では、スキーマは処理環境にインポートされる必要があります。例えば、XSLT には**xslt:import** 命令を使用してインポートします。

XML ノードからのスキーマ情報
関数 **altova:type** はXMLドキュメントのノードを送信しノードの型情報をPSVI から返します。

関数の名前指定と言語の適用性に関するメモ

Altova 拡張関数はXPath/XQuery 式で使用することができ、XPath、XQuery、およびXSLT 関数の標準ライブラリで使用可能な機能に更なる機能性を与えます。Altova 拡張関数はAltova **拡張関数名前空間**、**http://www.altova.com/xslt-extensions** に収められており、**altova:** プレフィックスが、このセクションでは使用されます。製品の今後のバージョンが拡張機能への継続的サポート、または個別の関数の振る舞いは変更する可能性があることご注意してください。Altova 拡張機能へのサポートに関しては、今後のリリースのドキュメントを参照してください。

XPath 関数 (XSLT 内のXPath 式で使用):	**XP1** **XP2** **XP3.1**
XSLT 関数 (XSLT 内のXPath 式で使用):	**XSLT1** **XSLT2** **XSLT3**
XQuery 関数 (XQuery 内のXQuery 式で使用):	**XQ1** **XQ3.1**

▼ Schema (引数無し)

altova:schema() as (function(xs:string) as item()*)? XP3.1 XQ3.1
schema コンポーネント全体を返します。**schema** コンポーネントを **schema** コンポーネントのプロパティの一つを選択して更にナビゲートすることができます。

- このプロパティがコンポーネントの場合、このコンポーネントのプロパティの1つを選択して更に深く他のステップをナビゲートすることができます。このステップは更にスキーマをナビゲートするために繰り返すことができます。
- コンポーネントが動的な値の場合、動的な値が返され更にナビゲートすることはできません。

schema コンポーネントのプロパティ:

```
"type definitions"
"attribute declarations"
"element declarations"
"attribute group definitions"
"model group definitions"
"notation declarations"
"identity-constraint definitions"
```

(**schema** 以外の)他のすべてのコンポーネント型のプロパティが下にリストされています。

メモ Xquery 式ではスキーマは明示的にインポートされる必要があります。XPath 式では、スキーマは処理環境にインポートされる必要があります。例えば、XSLT には**xslt:import** 命令を使用してインポートします。

⊟ *サンプル*

- `import schema "" at "C:\Test\ExpReport.xsd"; for $typedef in altova:schema() ("type definitions") return $typedef ("name")` はスキーマ内のすべての単純型または複合型の名前を返します
- `import schema "" at "C:\Test\ExpReport.xsd"; altova:schema() ("type definitions")[1]("name")` はスキーマ内の単純型または複合型の最初の名前を返します

コンポーネントとそのプロパティ

Assertion

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Assertion"
test	XPath プロパティレコード	

Attribute Declaration

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Attribute Declaration"
name	文字列	属性のローカル名
target namespace	文字列	属性の名前空間 URI
type definition	Simple Type または Complex Type	
scope	プロパティを持つ関数 (″class″:″Scope″, "variety": "global" or "local", "parent": the containing Complex Type or Attribute Group)	
value constraint	存在する場合、プロパティを持つ関数 (″class″: ″Value Constraint″、″variety″: ″fixed″ または″default″, ″value″: atomic value, ″lexical form″: string。″value″ プロパティは namespace-sensitive 型のために使用することはできないことに注意してください。	
inheritable	ブール値	

Attribute Group Declaration

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Attribute Group Definition"
name	文字列	属性グループのローカル名
target namespace	文字列	属性グループの名前空間 URI
attribute uses	(Attribute Use) のシーケンス	
attribute wildcard	任意の属性のワイルドカード	

Attribute Use

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Attribute Use"
required	ブール値	属性が必要な場合はtrue 、任意の場合はfalse
value constraint	Attribute Declaration を参照	
inheritable	ブール値	

Attribute Wildcard

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Wildcard"
namespace constraint	プロパティを持つ関数 (″class″: ″Namespace Constraint", "variety": "any"|"enumeration"|"not", "namespaces": sequence of xs:anyURI, "disallowed names": list containing QNames and/or the strings "defined" and "definedSiblings"	
process contents	文字列 (″strict″|″lax″|″skip″)	

Complex Type

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Complex Type"
name	文字列	(匿名の場合は空)型のローカル名
target namespace	文字列	(匿名の場合は空)型の名前空間 URI
base type definition	Complex Type Definition	
final	文字列のシーケンス ("restriction"|"extension")	
context	空のシーケンス(not implemented)	
derivation method	文字列 (″restriction″|″extension″)	
abstract	ブール値	
attribute uses	Attribute Use のシーケンス	
attribute wildcard	任意の属性のワイルドカード	
content type	プロパティを持つ関数: (″class″:″Content Type", "variety":string ("element-only"|"empty"|"mixed"|"simple"), particle: optional Particle, "open content": function with properties ("class":"Open Content", "mode": string ("interleave"|"suffix"), "wildcard": Wildcard), "simple type definition": Simple Type)	
prohibited	文字列のシーケンス	

substitutions	("restriction"|"extension")	
assertions	アサーションのシーケンス	

Element Declaration

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Complex Type"
name	文字列	(匿名の場合は空)型のローカル名
target namespace	文字列	(匿名の場合は空)型の名前空間 URI
type definition	Simple Type または Complex Type	
type table	プロパティを持つ関数 (″class″:″Type Table″, "alternatives": sequence of Type Alternative, "default type definition": Simple Type or Complex Type)	
scope	プロパティを持つ関数 (″class″:″Scope″, "variety": ("global"|"local"), "parent": optional Complex Type)	
value constraint	Attribute Declaration を参照	
nillable	ブール値	
identity-constraint definitions	Identity Constraint のシーケンス	
substitution group affiliations	Element Declaration のシーケンス	
substitution group exclusions	文字列のシーケンス ("restriction"|"extension")	
disallowed substitutions	文字列のシーケンス ("restriction"|"extension"|"substitution")	
abstract	ブール値	

Element Wildcard

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Wildcard"
namespace constraint	プロパティを持つ関数 (″class″: ″Namespace Constraint", "variety": "any"|"enumeration"|"not", "namespaces": sequence of xs:anyURI, "disallowed names": list containing QNames and/or the strings "defined" and "definedSiblings"	
process contents	文字列 (″strict″|″lax″|″skip″)	

Facet

プロパティ名	プロパティ型	プロパティ値

kind	文字列	ファセットの名前、例えば、"minLength" または "enumeration"
value	ファセットによる	ファセットの値
fixed	ブール値	
typed-value	ファセットの列挙のみ、Array(xs:anyAtomicType	列挙値を含む配列。それぞれがアトミック値のシーケンスである場合があります。(メモ列挙ファセットに関しては、実際の型にかかわらず "value" 値プロパティは文字列のシーケンスです)

Identity Constraint

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Identity-Constraint Definition"
name	文字列	制約のローカル名
target namespace	文字列	制約の名前空間 URI
identity-constraint category	文字列 ("key"|"unique"|"keyRef")	
selector	XPath プロパティレコード	
fields	XPath プロパティレコード のシーケンス	
referenced key	(keyRef のためのみ): Identity Constraint	対応するキー制約

Model Group

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Model Group"
compositor	文字列 ("sequence"|"choice"|"all")	
particles	Particle のシーケンス	

Model Group Definition

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Model Group Definition"
name	文字列	モデルグループのローカル名
target namespace	文字列	モデルグループの名前空間 URI
model group	Model Group	

Notation

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Notation Declaration"
name	文字列	表記のローカル名

target namespace	文字列	表記の名前空間 URI
system identifier	anyURI	
public identifier	文字列	

Particle

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Particle"
min occurs	整数	
max occurs	整数または文字列("unbounded")	
term	Element Declaration、Element Wildcard、または ModelGroup	

Simple Type

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Simple Type Definition"
name	文字列	(匿名の場合は空)型のローカル名
target namespace	文字列	(匿名の場合は空)型の名前空間 URI
final	文字列のシーケンス ("restriction"|"extension"|"list"|"union")	
context	含まれるコンポーネント	
base type definition	Simple Type	
facets	Sequence of Facet	
fundamental facets	(実装されていない)空のシーケンス	
variety	文字列 ("atomic"|"list"|"union")	
primitive type definition	Simple Type	
item type definition	(リスト型のみのための) Simple Type	
member type definitions	(ユニオン型のための) Simple Type のシーケンス	

Type Alternative

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Type Alternative"
test	XPath プロパティレコード	
type definition	Simple Type または Complex Type	

XPath Property Record

プロパティ名	プロパティ型	プロパティ値

namespace bindings	プロパティを持つ関数のシーケンス(″prefix″: string, "namespace": anyURI)	
default namespace	anyURI	
base URI	anyURI	XPath 式の静的ベースURI
expression	文字列	文字列としてのXPath 式

▼ Schema (2つの引数)

altova:schema(ComponentKind *as xs:string*, **Name** *as xs:QName*) **as (function(xs:string) as item()*)? XP3.1 XQ3.1**

2番目の引数内で与えられている名前と同じ名前を持つ最初の引数内で指定されているコンポーネントの型を返します。コンポーネントのプロパティの一つを選択して更にナビゲートすることができます。

- このプロパティがコンポーネントの場合、このコンポーネントのプロパティの1つを選択して更に深く他のステップをナビゲートすることができます。このステップは更にスキーマをナビゲートするために繰り返すことができます。
- コンポーネントが動的な値の場合、動的な値が返され更にナビゲートすることはできません。

メモ Xquery 式ではスキーマは明示的にインポートされる必要があります。XPath 式では、スキーマは処理環境にインポートされる必要があります。例えば、XSLT には**xslt:import** 命令を使用してインポートします。

⊟ *サンプル*

- ```
  import schema "" at "C:\Test\ExpReport.xsd";
  altova:schema("element declaration", xs:QName("OrgChart"))("type definition")
  ("content type")("particles")[3]!.("term")("kind")
  ```
  は3番目の`particles` コンポーネントの `kind` プロパティを返します。 この particles コンポーネントは `OrgChart` の `Qname` を持つ要素宣言の子孫です。

- ```
  import schema "" at "C:\Test\ExpReport.xsd";
  let $typedef := altova:schema("type definition", xs:QName("emailType"))
  for $facet in $typedef ("facets")
  return [$facet ("kind"), $facet("value")]
  ```
  は各 **emailType** コンポーネントの各 **facet** にそのファセットの型と値を含む列挙を返します。

コンポーネントとそのプロパティ

⊟ Assertion

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Assertion"
test	XPath プロパティレコード	

⊟ Attribute Declaration

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Attribute Declaration"
name	文字列	属性のローカル名

target namespace	文字列	属性の名前空間 URI
type definition	Simple Type または Complex Type	
scope	プロパティを持つ関数 (″class″:″Scope″, "variety": "global" or "local", "parent": the containing Complex Type or Attribute Group)	
value constraint	存在する場合、プロパティを持つ関数 (″class″: ″Value Constraint″、″variety″: ″fixed″ または″default″, ″value″: atomic value, ″lexical form″: string。″value″ プロパティは namespace-sensitive 型のために使用することはできないことに注意してください。	
inheritable	ブール値	

Attribute Group Declaration

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Attribute Group Definition"
name	文字列	属性グループのローカル名
target namespace	文字列	属性グループの名前空間 URI
attribute uses	(Attribute Use) のシーケンス	
attribute wildcard	任意の属性のワイルドカード	

Attribute Use

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Attribute Use"
required	ブール値	属性が必要な場合は true 、任意の場合は false
value constraint	Attribute Declaration を参照	
inheritable	ブール値	

Attribute Wildcard

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Wildcard"
namespace constraint	プロパティを持つ関数 (″class″: ″Namespace Constraint", "variety": "any"|"enumeration"|"not", "namespaces": sequence of xs:anyURI, "disallowed names": list containing QNames and/or the strings "defined" and "definedSiblings"	
process contents	文字列 (″strict″|″lax″|″skip″)	

(C) 2015-2021 Altova GmbH

Complex Type

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Complex Type"
name	文字列	(匿名の場合は空)型のローカル名
target namespace	文字列	(匿名の場合は空)型の名前空間 URI
base type definition	Complex Type Definition	
final	文字列のシーケンス ("restriction"|"extension")	
context	空のシーケンス(not implemented)	
derivation method	文字列 ("restriction"|"extension")	
abstract	ブール値	
attribute uses	Attribute Use のシーケンス	
attribute wildcard	任意の属性のワイルドカード	
content type	プロパティを持つ関数: ("class":"Content Type", "variety":string ("element-only"|"empty"|"mixed"|"simple"), particle: optional Particle, "open content": function with properties ("class":"Open Content", "mode": string ("interleave"|"suffix"), "wildcard": Wildcard), "simple type definition": Simple Type)	
prohibited substitutions	文字列のシーケンス ("restriction"|"extension")	
assertions	アサーションのシーケンス	

Element Declaration

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Complex Type"
name	文字列	(匿名の場合は空)型のローカル名
target namespace	文字列	(匿名の場合は空)型の名前空間 URI
type definition	Simple Type または Complex Type	
type table	プロパティを持つ関数 ("class":"Type Table", "alternatives": sequence of Type Alternative, "default type definition": Simple Type or Complex Type)	
scope	プロパティを持つ関数 ("class":"Scope", "variety": ("global"|"local"), "parent": optional Complex Type)	
value constraint	Attribute Declaration を参照	
nillable	ブール値	
identity-constraint	Identity Constraint のシーケンス	

definitions		
substitution group affiliations	Element Declaration のシーケンス	
substitution group exclusions	文字列のシーケンス ("restriction"|"extension")	
disallowed substitutions	文字列のシーケンス ("restriction"|"extension"|"substitution")	
abstract	ブール値	

Element Wildcard

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Wildcard"
namespace constraint	プロパティを持つ関数 (″class″: ″Namespace Constraint", "variety": "any"|"enumeration"|"not", "namespaces": sequence of xs:anyURI, "disallowed names": list containing QNames and/or the strings "defined" and "definedSiblings"	
process contents	文字列 (″strict″|″lax″|″skip″)	

Facet

プロパティ名	プロパティ型	プロパティ値
kind	文字列	ファセットの名前、例えば、″minLength″ または″enumeration″
value	ファセットによる	ファセットの値
fixed	ブール値	
typed-value	ファセットの列挙のみ、Array(xs:anyAtomicType	列挙値を含む配列。それぞれがアトミック値のシーケンスである場合があります。(メモ 列挙ファセットに関しては、実際の型にかかわらず″value″ 値プロパティは文字列のシーケンスです)

Identity Constraint

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Identity-Constraint Definition"
name	文字列	制約のローカル名
target namespace	文字列	制約の名前空間 URI
identity-constraint category	文字列 (″key″|″unique″|″keyRef″)	
selector	XPath プロパティレコード	
fields	XPath プロパティレコード のシーケンス	

(C) 2015-2021 Altova GmbH

referenced key	(keyRef のためのみ): Identity Constraint	対応するキー制約

Model Group

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Model Group"
compositor	文字列 ("sequence"|"choice"|"all")	
particles	Particle のシーケンス	

Model Group Definition

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Model Group Definition"
name	文字列	モデルグループのローカル名
target namespace	文字列	モデルグループの名前空間 URI
model group	Model Group	

Notation

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Notation Declaration"
name	文字列	表記のローカル名
target namespace	文字列	表記の名前空間 URI
system identifier	anyURI	
public identifier	文字列	

Particle

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Particle"
min occurs	整数	
max occurs	整数または文字列("unbounded")	
term	Element Declaration、Element Wildcard、または ModelGroup	

Simple Type

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Simple Type Definition"
name	文字列	(匿名の場合は空)型のローカル名
target namespace	文字列	(匿名の場合は空)型の名前空間 URI
final	文字列のシーケンス	

	("restriction"|"extension"|"list"|"union")	
context	含まれるコンポーネント	
base type definition	Simple Type	
facets	Sequence of Facet	
fundamental facets	(実装されていない)空のシーケンス	
variety	文字列 ("atomic"|"list"|"union")	
primitive type definition	Simple Type	
item type definition	(リスト型のみのための) Simple Type	
member type definitions	(ユニオン型のための) Simple Type のシーケンス	

Type Alternative

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Type Alternative"
test	XPath プロパティレコード	
type definition	Simple Type または Complex Type	

XPath Property Record

プロパティ名	プロパティ型	プロパティ値
namespace bindings	プロパティを持つ関数のシーケンス("prefix": string, "namespace": anyURI)	
default namespace	anyURI	
base URI	anyURI	XPath 式の静的ベース URI
expression	文字列	文字列としての XPath 式

型

altova:type(Node *as item?*) **as (function(xs:string) as item()*)?** **XP3.1** **XQ3.1**
関数 **altova:type** は XML ドキュメントの要素または属性ノードを送信しノードの型情報を PSVI から返します。

メモ スキーマが参照可能になるために XML ドキュメントはスキーマ宣言を持つ必要があります。

サンプル

- ```
  for $element in //Email
  let $type := altova:type($element)
  return $type
  ```
  はノードの型情報を含む関数を返します。

- ```
  for $element in //Email
  let $type := altova:type($element)
  return $type ("kind")
  ```


(C) 2015-2021 Altova GmbH

はノードの型のコンポーネント（単純型または複合型）を取りコンポーネントの`kind` property の値を返します。

コンポーネントとそのプロパティ

- Assertion

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Assertion"
test	XPath プロパティレコード	

- Attribute Declaration

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Attribute Declaration"
name	文字列	属性のローカル名
target namespace	文字列	属性の名前空間 URI
type definition	Simple Type または Complex Type	
scope	プロパティを持つ関数（"class":"Scope", "variety": "global" or "local", "parent": the containing Complex Type or Attribute Group)	
value constraint	存在する場合、プロパティを持つ関数（"class": "Value Constraint"、"variety": "fixed" または "default", "value": atomic value, "lexical form": string。"value" プロパティは namespace-sensitive 型のために使用することはできないことに注意してください。	
inheritable	ブール値	

- Attribute Group Declaration

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Attribute Group Definition"
name	文字列	属性グループのローカル名
target namespace	文字列	属性グループの名前空間 URI
attribute uses	(Attribute Use) のシーケンス	
attribute wildcard	任意の属性のワイルドカード	

- Attribute Use

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Attribute Use"
required	ブール値	属性が必要な場合は true 、任意の場合は false

value constraint	Attribute Declaration を参照	
inheritable	ブール値	

Attribute Wildcard

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Wildcard"
namespace constraint	プロパティを持つ関数 (″class″: ″Namespace Constraint", "variety": "any"|"enumeration"|"not", "namespaces": sequence of xs:anyURI, "disallowed names": list containing QNames and/or the strings "defined" and "definedSiblings"	
process contents	文字列 (″strict″|″lax″|″skip″)	

Complex Type

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Complex Type"
name	文字列	(匿名の場合は空)型のローカル名
target namespace	文字列	(匿名の場合は空)型の名前空間 URI
base type definition	Complex Type Definition	
final	文字列のシーケンス ("restriction"|"extension")	
context	空のシーケンス(not implemented)	
derivation method	文字列 (″restriction″|″extension″)	
abstract	ブール値	
attribute uses	Attribute Use のシーケンス	
attribute wildcard	任意の属性のワイルドカード	
content type	プロパティを持つ関数: (″class″:″Content Type", "variety":string ("element-only"|"empty"|"mixed"|"simple"), particle: optional Particle, "open content": function with properties ("class":"Open Content", "mode": string ("interleave"|"suffix"), "wildcard": Wildcard), "simple type definition": Simple Type)	
prohibited substitutions	文字列のシーケンス ("restriction"|"extension")	
assertions	アサーションのシーケンス	

Element Declaration

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Complex Type"
name	文字列	(匿名の場合は空)型のローカル名
target namespace	文字列	(匿名の場合は空)型の名前空間 URI
type definition	Simple Type または Complex Type	
type table	プロパティを持つ関数 (″class″:″Type Table″, "alternatives": sequence of Type Alternative, "default type definition": Simple Type or Complex Type)	
scope	プロパティを持つ関数 (″class″:″Scope″, "variety": ("global"|"local"), "parent": optional Complex Type)	
value constraint	Attribute Declaration を参照	
nillable	ブール値	
identity-constraint definitions	Identity Constraint のシーケンス	
substitution group affiliations	Element Declaration のシーケンス	
substitution group exclusions	文字列のシーケンス ("restriction"|"extension")	
disallowed substitutions	文字列のシーケンス ("restriction"|"extension"|"substitution")	
abstract	ブール値	

- Element Wildcard

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Wildcard"
namespace constraint	プロパティを持つ関数 (″class″: ″Namespace Constraint", "variety": "any"|"enumeration"|"not", "namespaces": sequence of xs:anyURI, "disallowed names": list containing QNames and/or the strings "defined" and "definedSiblings"	
process contents	文字列 (″strict″|″lax″|″skip″)	

- Facet

プロパティ名	プロパティ型	プロパティ値
kind	文字列	ファセットの名前、例えば、″minLength″ または″enumeration″
value	ファセットによる	ファセットの値
fixed	ブール値	

typed-value	ファセットの列挙のみ、Array(xs:anyAtomicType	列挙値を含む配列。それぞれがアトミック値のシーケンスである場合があります。（メモ列挙ファセットに関しては、実際の型にかかわらず "value" 値プロパティは文字列のシーケンスです）

Identity Constraint

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Identity-Constraint Definition"
name	文字列	制約のローカル名
target namespace	文字列	制約の名前空間 URI
identity-constraint category	文字列（"key"|"unique"|"keyRef"）	
selector	XPath プロパティレコード	
fields	XPath プロパティレコード のシーケンス	
referenced key	(keyRef のためのみ): Identity Constraint	対応するキー制約

Model Group

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Model Group"
compositor	文字列（"sequence"|"choice"|"all"）	
particles	Particle のシーケンス	

Model Group Definition

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Model Group Definition"
name	文字列	モデルグループのローカル名
target namespace	文字列	モデルグループの名前空間 URI
model group	Model Group	

Notation

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Notation Declaration"
name	文字列	表記のローカル名
target namespace	文字列	表記の名前空間 URI
system identifier	anyURI	
public identifier	文字列	

Particle

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Particle"
min occurs	整数	
max occurs	整数または文字列("unbounded")	
term	Element Declaration、Element Wildcard、または ModelGroup	

Simple Type

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Simple Type Definition"
name	文字列	(匿名の場合は空)型のローカル名
target namespace	文字列	(匿名の場合は空)型の名前空間 URI
final	文字列のシーケンス ("restriction"|"extension"|"list"|"union")	
context	含まれるコンポーネント	
base type definition	Simple Type	
facets	Sequence of Facet	
fundamental facets	(実装されていない)空のシーケンス	
variety	文字列 ("atomic"|"list"|"union")	
primitive type definition	Simple Type	
item type definition	(リスト型のみのための) Simple Type	
member type definitions	(ユニオン型のための) Simple Type のシーケンス	

Type Alternative

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Type Alternative"
test	XPath プロパティレコード	
type definition	Simple Type または Complex Type	

XPath Property Record

プロパティ名	プロパティ型	プロパティ値
namespace bindings	プロパティを持つ関数のシーケンス("prefix": string, "namespace": anyURI)	
default namespace	anyURI	
base URI	anyURI	XPath 式の静的ベース URI
expression	文字列	文字列としての XPath 式

26.2.1.7 XPath/XQuery 関数: シーケンス

Altova のシーケンス拡張関数はXPath とXQuery 式で使用することができ、XML スキーマの異なる日付および時刻データ型で保存されているデータを処理するための追加機能を提供します。このセクションの関数は、Altova の**XPath 3.0** および**XQuery 3.0** エンジンで使用することができます。これらの関数は、XPath/XQuery コンテキストで使用することができます。

関数の名前指定と言語の適用性に関するメモ

Altova 拡張関数はXPath/XQuery 式で使用することができ、XPath、XQuery、およびXSLT 関数の標準ライブラリで使用可能な機能に更なる機能性を与えます。Altova 拡張関数はAltova **拡張関数名前空間**、**http://www.altova.com/xslt-extensions** に収められており、**altova:** プレフィックスが、このセクションでは使用されます。製品の今後のバージョンが拡張機能への継続的サポート、または個別の関数の振る舞いは変更する可能性があることに注意してください。Altova 拡張機能へのサポートに関しては、今後のリリースのドキュメントを参照してください。

XPath 関数 (XSLT 内のXPath 式で使用):	XP1 XP2 XP3.1
XSLT 関数 (XSLT 内のXPath 式で使用):	XSLT1 XSLT2 XSLT3
XQuery 関数 (XQuery 内のXQuery 式で使用):	XQ1 XQ3.1

attributes [altova:]

altova:attributes(AttributeName *as xs:string*) を**attribute()*** とする XP3.1 XQ3.1
入力引数 AttributeName 内で与えられた名前と同じローカル名を持つすべての属性を返します。検索は大文字と小文字を区別し、attribute:: 軸に対して行われます。これは、コンテキストノードが親要素ノードである必要があることを意味します。

サンプル

- **altova:attributes**("MyAttribute") はMyAttribute()* を返します。

altova:attributes(AttributeName *as xs:string*, **SearchOptions** *as xs:string*) **asattribute()*** XP3.1 XQ3.1
入力引数 AttributeName 内で与えられた名前と同じローカル名を持つすべての属性を返します。検索は大文字と小文字を区別し、attribute:: 軸に対して行われます。コンテキストノードが親要素ノードである必要があります。第 2 引数はオプションのフラッグを含みます。使用することのできるフラッグは以下の通りです:
r = 正規表現検索を切り替えます; AttributeName は正規表現検索文字列である必要があります;
f = このオプションが指定されている場合、AttributeName は完全一致を提供します。それ以外の場合は、AttributeName は属性名に部分的に一致するとその属性を返します。例えば **f** が指定されていない場合、MyAtt は MyAttribute を返します。
i = 大文字と小文字を一致させる検索に切り替えます。
p = 検索に名前空間プレフィックスを含みます。AttributeName は、名前空間プレフィックスを含みます。例えば altova:MyAttribute。
フラッグは順序に関わりなく書き込むことができます。無効なフラッグはエラーを生成します。1 つまたは複数のフラッグを省略することができます。空の文字列は許可されていますが、引数を1 つしか持たない関数と同じ効果を持ちます(*前の署名*)、ですが、空のシーケンスは第 2 の引数としては許可されていません。

サンプル

- **altova:attributes**("MyAttribute", "rfip") はMyAttribute()* を返します。

- **altova:attributes**("MyAttribute", "pri") はMyAttribute()* を返します。
- **altova:attributes**("MyAtt", "rip") はMyAttribute()* を返します。
- **altova:attributes**("MyAttributes", "rfip") は不一致を返します。
- **altova:attributes**("MyAttribute", "") はMyAttribute()* を返します。
- **altova:attributes**("MyAttribute", "Rip") 認識されないフラッグエラーを返します。
- **altova:attributes**("MyAttribute",) 見つからない第 2 引数を返します。

elements [altova:]

altova:elements(ElementName *as xs:string*) を**element()*** とする **XP3.1** **XQ3.1**
入力引数 ElementName で与えられた名前と同じローカル名を持つすべての要素を返します。検索は大文字と小文字を区別して child:: 軸に対して実行されます。コンテキストノードは、検索される要素の親ノードである必要があります。

サンプル

- **altova:elements**("MyElement") はMyElement()* を返します。

altova:elements(ElementName *as xs:string*, **SearchOptions** *as xs:string*) **aselement()*** **XP3.1** **XQ3.1**
入力引数 ElementName 内で与えられた名前と同じローカル名を持つすべての属性を返します。検索は大文字と小文字を区別し、child:: 軸に対して行われます。コンテキストノードが親要素ノードである必要があります。第 2 引数はオプションのフラッグを含みます。使用することのできるフラッグは以下の通りです:
r = 正規表現検索を切り替えます; ElementName は正規表現検索文字列である必要があります;
f = このオプションが指定されている場合、ElementName は完全一致を提供します。それ以外の場合は、ElementName は属性名に部分的に一致するとその属性を返します。例えば **f** が指定されていない場合、MyElem はMyElement を返します。
i = 大文字と小文字を一致させる検索に切り替えます。
p = 検索に名前空間プレフィックスを含みます。ElementName は、名前空間プレフィックスを含みます。例えば altova:MyElement。
フラッグは順序に関わりなく書き込むことができます。無効なフラッグはエラーを生成します。1 つまたは複数のフラッグを省略することができます。空の文字列は許可されていますが、引数を1 つしか持たない関数と同じ効果を持ちます(*前の署名*)、ですが、空のシーケンスは第 2 の引数としては許可されていません。

サンプル

- **altova:elements**("MyElement", "rip") はMyElement()* を返します。
- **altova:elements**("MyElement", "pri") はMyElement()* を返します。
- **altova:elements**("MyElement", "") はMyElement()* を返します。
- **altova:attributes**("MyElem", "rip") はMyElement()* を返します。
- **altova:attributes**("MyElements", "rfip") 不一致を課wします。
- **altova:elements**("MyElement", "Rip") 認識されないフラッグエラーを返します。
- **altova:elements**("MyElement",) 見つからない第 2 引数を返します。

find-first [altova:]

altova:find-first((Sequence *as item()**), **(Condition(Sequence-Item** *as xs:boolean*)) を **item()?** とする **XP3.1** **XQ3.1**
この関数は2 つの引数を必要とします。最初の引数は1 つ、または1 つ以上のデータ型のアイテムのシーケンスです。第 2 の引数 Condition は、(1 のアリティを持つ) 1 つの引数 を必要とし、boolean を返す XPath 関数に対する参照です。
Condition で参照された関数の代わりに、**Sequence** の各アイテムが提出されます。(*注意:* この関数は1 つの引数のみを必要とします。) **Condition** 内の関数に**true()**と評価させる最初の **Sequence** アイテムは、**altova:find-first**、反復の終了の結果として返されます。

サンプル

- **altova:find-first**(5 to 10, function($a) {$a mod 2 = 0}) は xs:integer 6 を返します。**Condition** 引数は、**$a** という名のインライン関数を宣言し、定義します。XPath 3.0 インライン関数 **function()** を参照します。**altova: Sequence** 引数内の各アイテムでは、**find-first** がパスされ、代わりに、**$a** を入力値とします。入力値は関数定義($a mod 2 = 0)内の条件に対してテストされます。この条件を満たす最初の入力値が **altova:find-first**（この場合は 6）の結果として返されます。

- **altova:find-first**((1 to 10), (function($a) {$a+3=7})) は xs:integer 4 を返します。

更なるサンプル
ファイル **C:\Temp\Customers.xml** が存在する場合:

- **altova:find-first**(("C:\Temp\Customers.xml", "http://www.altova.com/index.html"), (doc-available#1)) は xs:string C:\Temp\Customers.xml を返します。

ファイル **C:\Temp\Customers.xml** が存在せず、**http://www.altova.com/index.html** が存在する場合:

- **altova:find-first**(("C:\Temp\Customers.xml", "http://www.altova.com/index.html"), (doc-available#1)) は xs:string http://www.altova.com/index.html を返します。

ファイル **C:\Temp\Customers.xml** が存在せず、**http://www.altova.com/index.html** も存在しない場合:

- **altova:find-first**(("C:\Temp\Customers.xml", "http://www.altova.com/index.html"), (doc-available#1)) 結果無しを返します。

上のサンプルについての注意点

- XPath 3.0 関数 doc-available は URI として使用され、ドキュメントノードが提出された URI で検出される場合 true を返する単一の引数を必要とします。（ですから、提出された URI でのドキュメントは XML ドキュメントである必要があります。）
- doc-available 関数は、altova:find-first の第 2 引数である、**Condition** で使用することができます。これは、1 つの引数（アリティ=1）のみを必要とするからであり、item() を入力（ URI として使用される文字列）として、boolean の値を返すからです。
- doc-available 関数は、参照されているだけで、呼び出されていない点に注意してください。アタッチされている #1 サフィックスは関数が 1 つのアリティであることを表示するためです。doc-available#1 の意味は以下のとおりです: アリティ=1 を持つ doc-availabe() 関数を使用し、最初のシーケンスの各アイテムの代わりに、単一引数として パスします。この結果、 2 つの文字列の各自つは、文字列をURIとして使用し、URIにドキュメントノードが存在するかテストする **doc-available()** にパスされます。1 つが各当する場合、**doc-available()** 関数は true() を評価し、シーケンス内のその文字列のインデックス ポジションは、 **altova:find-first** 関数の結果として返されます。doc-available() 関数に関しての注意点: 相対パスは、デフォルトで関数がロードされる XML ドキュメントの現在のベース URI に対して相対的に解決されます。

find-first-combination [altova:]

altova:find-first-combination((Seq-01 *as item()**), **(Seq-02** *as item()**), **(Condition(Seq-01-Item, Seq-02-Item** *as xs:boolean*)) を **item()*** とする **XP3.1** **XQ3.1**
この関数は 3 つの引数を必要とします:

- 最初の2 つの引数 Seq-01 とSeq-02, は、1 つまたは1 つ以上のデータ型のアイテムです。
- 第 3 の引数 Condition は、(2 のアリティを持つ) 2 つの引数を必要とし、boolean を返す XPath 関数に対しての参照です。

Seq-01 とSeq-02 のアイテムは指定された組み合わせ(各シーケンスからの1 つずつにアイテムで構成されるペア)で、Condition 内の関数の引数としてパスされました。組み合わせは以下のように指定されています。

```
If   Seq-01 = X1, X2, X3 ... Xn
And  Seq-02 = Y1, Y2, Y3 ... Yn
Then (X1 Y1), (X1 Y2), (X1 Y3) ... (X1 Yn), (X2 Y1), (X2 Y2) ... (Xn Yn)
```

Condition 関数にtrue() と評価するように指示する最初のペアはaltova:find-first-combination の結果として返されます。以下の点に注意してください: (i) Condition 関数が提出された引数ペア内で繰り返され、true()を評価しない場合、altova:find-first-combination は、結果を返しません; (ii) altova:find-first-combination の結果が常に(データ型の)アイテムのペアである場合またはアイテムでない場合。

サンプル

- altova:find-first-combination(11 to 20, 21 to 30, function($a, $b) {$a+$b = 32}) はxs:integers (11, 21) のシーケンスを返します。
- altova:find-first-combination(11 to 20, 21 to 30, function($a, $b) {$a+$b = 33}) はxs:integers (11, 22) のシーケンを返します。
- altova:find-first-combination(11 to 20, 21 to 30, function($a, $b) {$a+$b = 34}) は xs:integers (11, 23) のシーケンを返します。

find-first-pair [altova:]

altova:find-first-pair((Seq-01 *as item()**), (Seq-02 *as item()**), (Condition(Seq-01-Item, Seq-02-Item *as xs:boolean*)) をitem()* とする XP3.1 XQ3.1
この関数は3 つの引数を必要とします:

- 最初の2 つの引数 Seq-01 とSeq-02, は、1 つまたは1 つ以上のデータ型のアイテムです。
- 第 3 の引数 Condition は、(2 のアリティを持つ) 2 つの引数を必要とし、boolean を返す XPath 関数に対しての参照です。

Seq-01 とSeq-02 のアイテムは指定された組み合わせで、Condition 内の関数の引数としてパスされました。組み合わせは以下のように指定されています。

```
If   Seq-01 = X1, X2, X3 ... Xn
And  Seq-02 = Y1, Y2, Y3 ... Yn
Then (X1 Y1), (X2 Y2), (X3 Y3) ... (Xn Yn)
```

Condition 関数にtrue() と評価するように指示する最初のペアはaltova:find-first-pair の結果として返されます。以下の点に注意してください: (i) Condition 関数が提出された引数ペア内で繰り返され、true()を評価しない場合、altova:find-first-pair は、結果を返しません; (ii) altova:find-first-combination の結果が常に(データ型の) アイテムのペアである場合またはアイテムでない場合。

サンプル

- altova:find-first-pair(11 to 20, 21 to 30, function($a, $b) {$a+$b = 32}) は xs:integers (11, 21) のシーケンスを返します。
- altova:find-first-pair(11 to 20, 21 to 30, function($a, $b) {$a+$b = 33}) は*結果無し*を返します。

上の2 つのサンプルに表示される通り、ペアの順序は以下の通りです: (11, 21) (12, 22) (13, 23)...(20,

30) 。(33 を返す指示されたペアがないため) この理由で第 2 のサンプルは *結果無し*を返します。

find-first-pair-pos [altova:]

altova:find-first-pair-pos((Seq-01 *as item()**)**, (Seq-02** *as item()**)**, (Condition(Seq-01-Item, Seq-02-Item** *as xs:boolean*)**)** を**xs:integer** とする **XP3.1** **XQ3.1**
この関数は3 つの引数を必要とします:

- 最初の2 つの引数 **Seq-01** と**Seq-02**, は、1 つまたは1 つ以上のデータ型のアイテムです。
- 第 3 の引数 **Condition** は、(2 のアリティを持つ) 2 つの引数を必要とし、boolean を返す XPath 関数に対しての参照です。

Seq-01 と**Seq-02** のアイテムは指定された組み合わせで、**Condition** 内の関数の引数として パスされました。組み合わせは以下のように指定されています。

```
If   Seq-01 = X1, X2, X3 ... Xn
And  Seq-02 = Y1, Y2, Y3 ... Yn
Then (X1 Y1), (X2 Y2), (X3 Y3) ... (Xn Yn)
```

Condition 関数に**true()**を評価させる、最初に指示されたペアのインデックス ポジションは、は**altova:find-first-pair-pos** の結果として返されます。関数が提出された引数ペア内で繰り返され、**true()**を一度も評価しない場合、**altova:find-first-pair-pos** *結果無し*が返します。

サンプル

- **altova:find-first-pair-pos**(11 to 20, 21 to 30, function($a, $b) {$a+$b = 32}) は 1 を返します
- **altova:find-first-pair-pos**(11 to 20, 21 to 30, function($a, $b) {$a+$b = 33}) は*結果無し*を返します。

上の2 つのサンプルに表示される通り、ペアの順序は以下の通りです: (11, 21) (12, 22) (13, 23)...(20, 30)。最初のサンプルでは、最初のペアは**Condition** 関数に**true()**を評価させ、シーケンス内のインデックス ポジションには**1** が返されます。第 2 のサンプルでは、 33 を返すペアがないため、*結果無し*が返します。

find-first-pos [altova:]

altova:find-first-pos((Sequence *as item()**)**, (Condition(Sequence-Item** *as xs:boolean*)**)** を**xs:integer** とする **XP3.1** **XQ3.1**
この関数は2 つの引数を必要とします。最初の引数は1 つ、または1 つ以上のデータ型のアイテムのシーケンスです。第 2 の引数 Condition は、(1 のアリティを持つ) 1 つの引数 を必要とし、boolean を返す XPath 関数に対する参照です。Condition で参照された関数の代わりに、**Sequence** の各アイテムが提出されます。(*注意:* この関数は1 つの引数のみを必要とします。) **Condition** 内の関数に**true()**と評価させる最初の **Sequence** アイテムは、**altova:find-first-pos** の結果として返された**Sequence** 内インデックス ポジションを持ちます。

サンプル

- **altova:find-first-pos**(5 to 10, function($a) {$a mod 2 = 0}) は**xs:integer** 2 を返します。
  Condition 引数は、**$a** という名のインライン関数を宣言し、定義します。XPath 3.0 インライン関数 **function()**を参照します。 **Sequence** 引数内の各アイテムでは、**find-first-pos** が パスされ、代わりに、**$a** を入力値とします。入力値は関数定義($a mod 2 = 0)内の条件に対してテストされます。この条件を満たす最初の入力値が **altova:find-first-pos**((シーケンス内で条件を満たす最初の値である6 がシーケンスのインデックス位置 2 にあるためこの場合は2、) の結果として返されます。

- **altova:find-first-pos**((2 to 10), (function($a) {$a+3=7})) は xs:integer 3 を返します。

更なるサンプル
ファイル **C:\Temp\Customers.xml** が存在する場合:

- **altova:find-first-pos**(("C:\Temp\Customers.xml", "http://www.altova.com/index.html"), (doc-available#1)) は 1 を返します。

ファイル **C:\Temp\Customers.xml** が存在せず、**http://www.altova.com/index.html** が存在する場合:

- **altova:find-first-pos**(("C:\Temp\Customers.xml", "http://www.altova.com/index.html"), (doc-available#1)) は 2 を返します。

ファイル **C:\Temp\Customers.xml** が存在せず、**http://www.altova.com/index.html** も存在しない場合:

- **altova:find-first-pos**(("C:\Temp\Customers.xml", "http://www.altova.com/index.html"), (doc-available#1)) 結果無しを返します。

上のサンプルについての注意点

- XPath 3.0 関数 doc-available は URI として使用され、ドキュメントノードが提出された URI で検出される場合 true を返する単一の引数を必要とします。(ですから、提出された URI でのドキュメントは XML ドキュメントである必要があります。)
- doc-available 関数は、altova:find-first-pos の第 2 引数である、**Condition** で使用することができます。これは、1 つの引数 (アリティ=1) のみを必要とするからであり、item() を入力(URI として使用される文字列) として、boolean の値を返すからです。
- doc-available 関数は、参照されているだけで、呼び出されていない点に注意してください。アタッチされている #1 サフィックスは関数が 1 つのアリティであることを表示するためです。doc-available#1 の意味は以下のとおりです: アリティ=1 を持つ doc-availabe() 関数を使用し、最初のシーケンスの各アイテムの代わりに、単一引数として パスします。 この結果、 2 つの文字列の各自つは、文字列をURIとして使用し、URIにドキュメントノードが存在するかテストする **doc-available()** にパスされます。1 つが各当する場合、**doc-available()** 関数は true() を評価し、シーケンス内のその文字列のインデックス ポジションは、 **altova:find-first-pos** 関数の結果として返されます。doc-available() 関数に関しての注意点: 相対パスは、デフォルトで関数がロードされる XML ドキュメントの現在のベース URI に対して相対的に解決されます。

for-each-attribute-pair [altova:]

altova:for-each-attribute-pair(Seq1 *as element()?*, **Seq2** *as element()?*, **Function** *as function()*) **asitem()*** **XP3.1** **XQ3.1**
ペアの一つの属性が最初の要素を取得し、もう一つの属性が2番目の要素から取得された箇所で、最初の2つの引数は、2つの要素、および、属性ペアを作成するために使用された属性を識別します。要素ペアは同じ名前を持つことをベースに選択され、ペアは名前別に、アルファベット順にセットに並べ替えられます。属性のために他の要素上に対応する属性が存在しない場合、ペアのジョイントは解除されます。これは、一つのメンバーにより飲み構成されることを意味します。関数アイテム(3番目の引数 Function) はアイテムのシーケンスである出力であるペアのシーケンス内のペア(ジョイント、ジョイントの解除) に個別に適用されます。

サンプル

- **altova:for-each-attribute-pair**(/Example/Test-A, /Example/Test-B, function($a, $b){$a+b}) は以下 を返します。...

```
(2, 4, 6) if
<Test-A att1="1" att2="2" att3="3" />
<Test-B att1="1" att2="2" att3="3" />

(2, 4, 6) if
<Test-A att2="2" att1="1" att3="3" />
<Test-B att3="3" att2="2" att1="1" />

(2, 6) if
<Test-A att4="4" att1="1" att3="3" />
<Test-B att3="3" att2="2" att1="1" />
```

メモ: 結果 (2, 6) は次のアクションにより取得されます: (1+1, ()+2, 3+3, 4+()) 。演算子の一つが空のシーケンスの場合、アイテム 2 と 4 の場合同様、結果は空のシーケンスになります。

- **altova:for-each-attribute-pair**(/Example/Test-A, /Example/Test-B, concat#2) は以下を返します。

```
(11, 22, 33) if
<Test-A att1="1" att2="2" att3="3" />
<Test-B att1="1" att2="2" att3="3" />

(11, 2, 33, 4) if
<Test-A att4="4" att1="1" att3="3" />
<Test-B att3="3" att2="2" att1="1" />
```

for-each-combination [altova:]

altova:for-each-combination(FirstSequence *as item()**, **SecondSequence** *as item()**, **function($i,$j){$i || $j}) を item()* とする** **XP3.1** **XQ3.1**

第 3 の引数として与えられた、インライン関数を返します。FirstSequence 内の各 $i を SecondSequence内の各 $j と結合します。出力はこれらのアイテムのシーケンスです。

サンプル

- **altova:for-each-combination**(('a', 'b', 'c'), ('1', '2', '3'), function($i, $j) {$i || $j}) は ('a1', 'a2', 'a3', 'b1', 'b2', 'b3', 'c1', 'c2', 'c3') を返します。

for-each-combination [altova:]

altova:for-each-combination(FirstSequence *as item()**, **SecondSequence** *as item()**, **Function($i,$j){$i || $j}) as item()*** **XP3.1** **XQ3.1**

最初の2 つの引数内の2 つのシーケンスのアイテムは最初のシーケンスの各アイテムが2 番目のシーケンスになるように結合されるように結合されます。3 番目の引数として与えられる関数は結果のシーケンス内の組み合わせに適用され、アイテムのシーケンスである出力内で出力されます（*サンプル参照*）。

サンプル

- **altova:for-each-combination**(('a', 'b', 'c'), ('1', '2', '3'), function($i, $j) {$i || $j}) は ('a1', 'a2', 'a3', 'b1', 'b2', 'b3', 'c1', 'c2', 'c3') を返します。

▼ substitute-empty [altova:]

altova:substitute-empty(FirstSequence *as item()**, **SecondSequence** *as item()*) を **item()*** とする **XP3.1** **XQ3.1**
FirstSequence が空の場合、SecondSequence を返します。FirstSequence が空でない場合、FirstSequence を返します。

サンプル

- **altova:substitute-empty**((1,2,3), (4,5,6)) は (1,2,3) を返します。
- **altova:substitute-empty**((), (4,5,6)) は (4,5,6) を返します。

26.2.1.8 XPath/XQuery 関数: 文字列

Altova の文字列拡張関数は XPath と XQuery 式で使用することができ、XML スキーマの異なる日付および時刻データ型で保存されているデータを処理するための追加機能を提供します。このセクションの関数は、Altova の **XPath 3.0** および **XQuery 3.0** エンジンで使用することができます。これらの関数は、XPath/XQuery コンテキストで使用することができます。

関数の名前指定と言語の適用性に関するメモ

Altova 拡張関数は XPath/XQuery 式で使用することができ、XPath、XQuery、および XSLT 関数の標準ライブラリで使用可能な機能に更なる機能性を与えます。Altova 拡張関数は **Altova 拡張関数名前空間**、**http://www.altova.com/xslt-extensions** に収められており、**altova:** プレフィックスが、このセクションでは使用されます。製品の今後のバージョンが拡張機能への継続的サポート、または個別の関数の振る舞いは変更する可能性があることに注意してください。Altova 拡張機能へのサポートに関しては、今後のリリースのドキュメントを参照してください。

XPath 関数 (XSLT 内の XPath 式で使用):	**XP1** **XP2** **XP3.1**
XSLT 関数 (XSLT 内の XPath 式で使用):	**XSLT1** **XSLT2** **XSLT3**
XQuery 関数 (XQuery 内の XQuery 式で使用):	**XQ1** **XQ3.1**

▼ camel-case [altova:]

altova:camel-case(InputString *as xs:string*) を **xs:string** とする **XP3.1** **XQ3.1**
入力文字列を **InputString** をキャメルケースで返します。文字列は(空白スペースのショートカットである) 正規表現 **'\s'** を使用して分析されます。空白文字または連続する空白文字のシーケンスの後の最初の非空白スペース文字は大文字です。出力文字列の最初の文字は大文字です。

サンプル

- **altova:camel-case**("max") Max を返します。
- **altova:camel-case**("max max") Max Max を返します。
- **altova:camel-case**("file01.xml") File01.xml を返します。
- **altova:camel-case**("file01.xml file02.xml") File01.xml File02.xml を返します。
- **altova:camel-case**("file01.xml file02.xml") File01.xml File02.xml を返します。
- **altova:camel-case**("file01.xml -file02.xml") File01.xml -file02.xml を返します。

altova:camel-case(InputString *as xs:string*, **SplitChars** *as xs:string*, **IsRegex** *as xs:boolean*) を **xs:string** とする **XP3.1** **XQ3.1**
SplitChars を使用して、次の大文字をトリガーする文字を決定し、入力文字列を **InputString** キャメルケースに変換しま

す。 SplitChars は、IsRegex = true() の場合、または、IsRegex = false() の場合、プレーン文字は正規表現として使用されます。出力文字列の最初の文字は大文字です。

サンプル

- altova:camel-case("setname getname", "set|get", true()) setName getName を返します。
- altova:camel-case("altova\documents\testcases", "\", false()) Altova\Documents\Testcases を返します。

char [altova:]

altova:char(Position *as xs:integer*) を xs:string とする XP3.1 XQ3.1

xs:string.Position に対してのコンテキスト アイテムの値を変換することにより得られた文字列内の Position 引数により指定されたポジションにある文字を含む文字列を返します。引数により提出されたインデックスに文字が存在しない場合、結果文字列は空です。

サンプル

コンテキスト アイテムが 1234ABCD の場合:

- altova:char(2) は 2 を返します。
- altova:char(5) は A を返します。
- altova:char(9) は空の文字列を返します。
- altova:char(-2) は空の文字列を返します。

altova:char(InputString *as xs:string*, Position *as xs:integer*) を xs:string とする XP3.1 XQ3.1

引数として提出された文字列内の Position 引数により指定されたポジションでの文字を含む文字列を返します。Position 引数により提出されたインデックスに文字が存在しない場合、結果文字列は空です。

サンプル

- altova:char("2014-01-15", 5) は - を返します。
- altova:char("USA", 1) は U を返します。
- altova:char("USA", 10) は空の文字列を返します。
- altova:char("USA", -2) は空の文字列を返します。

create-hash-from-string[altova:]

altova:create-hash-from-string(InputString *as xs:string*) as xs:string XP2 XQ1 XP3.1 XQ3.1

altova:create-hash-from-string(InputString *as xs:string*, HashAlgo *as xs:string*) as xs:string XP2 XQ1 XP3.1 XQ3.1

HashAlgo 引数により指定されているハッシュアルゴリズムを使用して InputString からハッシュ文字列を生成します。次のハッシュアルゴリズムは(大文字、または、小文字で指定されている可能性があります: MD5, SHA-224, SHA-256, SHA-384, SHA-512。(最初の署名を参照してください) 2番目の引数が指定されていない場合、SHA-256 ハッシュアルゴリズムが使用されます。

サンプル

- altova:create-hash-from-string('abc') は SHA-256 ハッシュアルゴリズムを使用して生成されたハッシュ文字列 を返します。
- altova:create-hash-from-string('abc', 'md5') は MD5 ハッシュアルゴリズムを使用して生成されたハッシュ文字列 を返します。
- altova:create-hash-from-string('abc', 'MD5') は MD5 ハッシュアルゴリズムを使用して生成された

ハッシュ文字列 を返します。

first-chars [altova:]

altova:first-chars(X-Number *as xs:integer*) を **xs:string** とする **XP3.1** **XQ3.1**
xs:string に対するコンテキストアイテムの値を変換することにより得られた最初の X-Number 文字を含む文字列を返します。

サンプル

コンテキストアイテムが **1234ABCD** の場合:

- **altova:first-chars**(2) は 12 を返します。
- **altova:first-chars**(5) は 1234A を返します。
- **altova:first-chars**(9) は 1234ABCD を返します。

altova:first-chars(InputString *as xs:string*, **X-Number** *as xs:integer*) を **xs:string** とする **XP3.1** **XQ3.1**
InputString 引数として提出された文字列の最初の文字を含む文字列を返します。

サンプル

- **altova:first-chars**("2014-01-15", 5) は 2014- を返します。
- **altova:first-chars**("USA", 1) は U を返します。

format-string [altova:]

altova:format-string(InputString *as xs:string*, **FormatSequence** *as item()**) **as xs:string** **XP3.1** **XQ3.1**
入力文字列（最初の引数）には、配置パラメーター（%1, %2, etc）が含まれています。各パラメーターは、（2番目の引数として提出されている）フォーマットシーケンス内の対応するポジションでロケートされる文字列アイテムと置き換えられます。フォーマットシーケンス内の最初のアイテムは、配置パラメーター %1 を置換、二番目のアイテムは %2 を置き換えます。関数は、書式設定された代替を持つ文字列を返します。配置パラメーターに文字列が存在しない場合は、配置パラメーターが返されます。これは、配置パラメーターのインデックスが書式シーケンス内のアイテムの数よりも大きい場合発生します。

サンプル

- **altova:format-string**('Hello %1, %2, %3', ('Jane','John','Joe')) は "Hello Jane, John, Joe" を返します。
- **altova:format-string**('Hello %1, %2, %3', ('Jane','John','Joe', 'Tom')) は "Hello Jane, John, Joe" を返します。
- **altova:format-string**('Hello %1, %2, %4', ('Jane','John','Joe', 'Tom')) は "Hello Jane, John, Tom" を返します。
- **altova:format-string**('Hello %1, %2, %4', ('Jane','John','Joe')) は "Hello Jane, John, %4" を返します。

last-chars [altova:]

altova:last-chars(X-Number *as xs:integer*) を **xs:string** とする **XP3.1** **XQ3.1**
xs:string に対してのコンテキストアイテムの値の変換より取得された文字列の最後の X-Number 文字を含んでいる文字列を返します。

サンプル

コンテキストアイテムが **1234ABCD** の場合:

- **altova:last-chars**(2) は CD を返します。
- **altova:last-chars**(5) は 4ABCD を返します。
- **altova:last-chars**(9) は 1234ABCD を返します。

altova:last-chars(InputString *as xs:string*, **X-Number** *as xs:integer*) **as xs:string** とする
XP3.1 **XQ3.1**
引数として提出された文字列の最後の X-Number 文字を含んでいる文字列を返します。

サンプル

- **altova:last-chars**("2014-01-15", 5) は 01-15 を返します。
- **altova:last-chars**("USA", 10) は USA を返します。

pad-string-left [altova:]

altova:pad-string-left(StringToPad *as xs:string*, **StringLength** *as xs:integer*, **PadCharacter** *as xs:string*) を **xs:string** とする **XP3.1** **XQ3.1**
PadCharacter 引数は1文字です。文字列の左側にパッドされ、この数が、StringLength 引数の整数の値と等しくなるように StringToPad の文字の数を増やします。StringLength 引数は任意の整数の値（正数または負数）を持つことができますが、パディングは、StringLength の値が StringToPad 内の文字数より多い場合の場合のみ発生します。もし、StringToPad が StringLength の値より多くの文字数も持つ場合、StringToPad は変更されません。

サンプル

- **altova:pad-string-left**('AP', 1, 'Z') は 'AP' を返します。
- **altova:pad-string-left**('AP', 2, 'Z') は 'AP' を返します。
- **altova:pad-string-left**('AP', 3, 'Z') は 'ZAP' を返します。
- **altova:pad-string-left**('AP', 4, 'Z') は 'ZZAP' を返します。
- **altova:pad-string-left**('AP', -3, 'Z') は 'AP' を返します。
- **altova:pad-string-left**('AP', 3, 'YZ') は[パッド文字が長すぎます] エラーを返します。

pad-string-right [altova:]

altova:pad-string-right(StringToPad *as xs:string*, **StringLength** *as xs:integer*, **PadCharacter** *as xs:string*) を **xs:string** とする **XP3.1** **XQ3.1**
PadCharacter 引数は1文字です。文字列の右側にパッドされ、この数が、StringLength 引数の整数の値と等しくなるように StringToPad の文字の数を増やします。StringLength 引数は任意の整数の値（正数または負数）を持つことができますが、パディングは、StringLength の値が StringToPad 内の文字数より多い場合の場合のみ発生します。もし、StringToPad が StringLength の値より多くの文字数も持つ場合、StringToPad は変更されません。

サンプル

- **altova:pad-string-right**('AP', 1, 'Z') を 'AP' を返します。
- **altova:pad-string-right**('AP', 2, 'Z') を 'AP' を返します。
- **altova:pad-string-right**('AP', 3, 'Z') を 'APZ' を返します。
- **altova:pad-string-right**('AP', 4, 'Z') を 'APZZ' を返します。
- **altova:pad-string-right**('AP', -3, 'Z') を 'AP' を返します。
- **altova:pad-string-right**('AP', 3, 'YZ') は[パッド文字が長すぎます] エラーを返します。

repeat-string [altova:]

altova:repeat-string(InputString *as xs:string*, **Repeats** *as xs:integer*) を **xs:string** とする

XP2 XQ1 XP3.1 XQ3.1
最初の InputString 引数により構成される文字列、Repeats 回繰り返してを生成します。

サンプル

- altova:repeat-string("Altova #", 3) は "Altova #Altova #Altova #" を返します。

substring-after-last [altova:]

altova:substring-after-last(MainString *as xs:string*, CheckString *as xs:string*) を xs:string とする XP3.1 XQ3.1
CheckString が MainString 内で検出された場合、MainString 内の CheckString が発生した後のサブ文字列が返されます。MainString 内で CheckString が検出されない場合、空の文字列が返されます。CheckString が空の文字列の場合、MainString 全体が返されます。一度以上発生する場合、CheckString の最後の発生の後の サブ文字列が返されます。

サンプル

- altova:substring-after-last('ABCDEFGH', 'B') は 'CDEFGH' を返します。
- altova:substring-after-last('ABCDEFGH', 'BC') は 'DEFGH' を返します。
- altova:substring-after-last('ABCDEFGH', 'BD') は '' を返します。
- altova:substring-after-last('ABCDEFGH', 'Z') は '' を返します。
- altova:substring-after-last('ABCDEFGH', '') は 'ABCDEFGH' を返します。
- altova:substring-after-last('ABCD-ABCD', 'B') は 'CD' を返します。
- altova:substring-after-last('ABCD-ABCD-ABCD', 'BCD') は '' を返します。

substring-before-last [altova:]

altova:substring-before-last(MainString *as xs:string*, CheckString *as xs:string*) を xs:string とする XP3.1 XQ3.1
CheckString が MainString 内で検出された場合、MainString 内の CheckString が発生する前のサブ文字列が返されます。MainString 内で CheckString が一度以上発生する場合、CheckString の最後の発生の前の サブ文字列が返されます。

サンプル

- altova:substring-before-last('ABCDEFGH', 'B') は 'A' を返します。
- altova:substring-before-last('ABCDEFGH', 'BC') は 'A' を返します。
- altova:substring-before-last('ABCDEFGH', 'BD') は '' を返します。
- altova:substring-before-last('ABCDEFGH', 'Z') は '' を返します。
- altova:substring-before-last('ABCDEFGH', '') は '' を返します。
- altova:substring-before-last('ABCD-ABCD', 'B') は 'ABCD-A' を返します。
- altova:substring-before-last('ABCD-ABCD-ABCD', 'ABCD') は 'ABCD-ABCD-' を返します。

substring-pos [altova:]

altova:substring-pos(StringToCheck *as xs:string*, StringToFind *as xs:string*) を xs:integer とする XP3.1 XQ3.1
StringToCheck 内での StringToFind の最初の発生の文字位置を整数として返します。StringToCheck の最初の文字は、位置 1 にあります。StringToFind が StringToCheck 内で発生しない場合、整数 0 が返されます。第 2 またはその後の StringToCheck, 発生を確認するには、この関数の次の署名を確認してください。

サンプル

- altova:substring-pos('Altova', 'to') は 3 を返します。

- **altova:substring-pos**('Altova', 'tov') は 3 を返します。
- **altova:substring-pos**('Altova', 'tv') は 0 を返します。
- **altova:substring-pos**('AltovaAltova', 'to') は 3 を返します。

altova:substring-pos(StringToCheck *as xs:string*, **StringToFind** *as xs:string*, **Integer** *as xs:integer*) を **xs:integer** とする **XP3.1** **XQ3.1**
StringToCheck 内での StringToFind の最初の発生の文字位置を返します。Integer 引数により与えられた、文字位置から StringToFind の検索が開始されます。この位置の前の文字サブ文字列は検索されません。しかし、返された整数は、全体文字列 StringToCheck の検索された文字列の位置です。この署名は、StringToCheck 内で複数回発生する、第 2 または後の発生を検索する際に役に立ちます。 StringToFind が StringToCheck 内で発生しない場合、整数 0 が返されます。

サンプル

- **altova:substring-pos**('Altova', 'to', 1) は 3 を返します。
- **altova:substring-pos**('Altova', 'to', 3) は 3 を返します。
- **altova:substring-pos**('Altova', 'to', 4) は 0 を返します。
- **altova:substring-pos**('Altova-Altova', 'to', 0) は 3 を返します。
- **altova:substring-pos**('Altova-Altova', 'to', 4) は 10 を返します。

trim-string [altova:]

altova:trim-string(InputString *as xs:string*) を **xs:string** とする **XP3.1** **XQ3.1**
この関数は xs:string 引数を必要とし、先頭または後続の空白を削除し、トリミングされた xs:string を返します。

サンプル

- **altova:trim-string**(" Hello World ")) は "Hello World" を返します。
- **altova:trim-string**("Hello World ")) は "Hello World" を返します。
- **altova:trim-string**(" Hello World")) は "Hello World" を返します。
- **altova:trim-string**("Hello World")) は "Hello World" を返します。
- **altova:trim-string**("Hello World")) は "Hello World" を返します。

trim-string-left [altova:]

altova:trim-string-left(InputString *as xs:string*) を **xs:string** とする **XP3.1** **XQ3.1**
この関数は xs:string 引数を必要とし、先頭または後続の空白を削除し、トリミングされた xs:string を返します。

サンプル

- **altova:trim-string-left**(" Hello World ")) は "Hello World " を返します。
- **altova:trim-string-left**("Hello World ")) は "Hello World " を返します。
- **altova:trim-string-left**(" Hello World")) は "Hello World" を返します。
- **altova:trim-string-left**("Hello World")) は "Hello World" を返します。
- **altova:trim-string-left**("Hello World")) は "Hello World" を返します。

trim-string-right [altova:]

altova:trim-string-right(InputString *as xs:string*) を **xs:string** とする **XP3.1** **XQ3.1**
この関数は xs:string 引数を必要とし、先頭または後続の空白を削除し、トリミングされた xs:string を返します。

サンプル

- **altova:trim-string-right**(" Hello World ")) は" Hello World" を返します。
- **altova:trim-string-right**("Hello World ")) は"Hello World" を返します。
- **altova:trim-string-right**(" Hello World")) は" Hello World" を返します。
- **altova:trim-string-right**("Hello World")) は"Hello World" を返します。
- **altova:trim-string-right**("Hello World")) は"Hello World" を返します。

26.2.1.9 XPath/XQuery 関数: その他

XPath/XQuery 拡張関数の以下の一般的な目的は現在のXMLSpy バージョンでサポートされており(i) XSLT コンテキスト内のXPath 式 または(ii) Xquery ドキュメント内のXQuery 式内で使用することができます。。

関数の名前指定と言語の適用性に関するメモ

Altova 拡張関数はXPath/XQuery 式で使用することができ、XPath、XQuery、およびXSLT 関数の標準ライブラリで使用可能な機能に更なる機能性を与えます。Altova 拡張関数はAltova 拡張関数名前空間、**http://www.altova.com/xslt-extensions** に収められており、**altova:** プレフィックスが、このセクションでは使用されます。製品の今後のバージョンが拡張機能への継続的サポート、または個別の関数の振る舞いは変更する可能性があることに注意してください。Altova 拡張機能へのサポートに関しては、今後のリリースのドキュメントを参照してください。

XPath 関数 (XSLT 内のXPath 式で使用):	**XP1** **XP2** **XP3.1**
XSLT 関数 (XSLT 内のXPath 式で使用):	**XSLT1** **XSLT2** **XSLT3**
XQuery 関数 (XQuery 内のXQuery 式で使用):	**XQ1** **XQ3.1**

decode-string [altova:]

altova:decode-string(Input *as xs:base64Binary*) **as xs:string** **XP3.1** **XQ3.1**
altova:decode-string(Input *as xs:base64Binary*, **Encoding** *as xs:string*) **as xs:string** **XP3.1** **XQ3.1**
指定されたエンコードを使用して送信されたbase64Binary 入力を文字列にデコードします。エンコードが指定されていない場合UTF-8 エンコードが使用されます。以下のエンコードがサポートされます: US-ASCII, ISO-8859-1, UTF-16, UTF-16LE, UTF-16BE, ISO-10646-UCS2, UTF-32, UTF-32LE, UTF-32BE, ISO-10646-UCS4

サンプル

- **altova:decode-string**($XML1/MailData/Meta/b64B) はbase64Binary 入力をUTF-8 エンコード済み文字列として返します。
- **altova:decode-string**($XML1/MailData/Meta/b64B, "UTF-8") はbase64Binary 入力をUTF-8-エンコード済み文字列として返します。
- **altova:decode-string**($XML1/MailData/Meta/b64B, "ISO-8859-1") はbase64Binary 入力をISO-8859-1 エンコード済み文字列として返します。

encode-string [altova:]

altova:encode-string(InputString *as xs:string*) **as xs:base64Binaryinteger** **XP3.1** **XQ3.1**
altova:encode-string(InputString *as xs:string*, **Encoding** *as xs:string*) **as**

xs:base64Binaryinteger XP3.1 XQ3.1
与えらえている場合指定されているエンコードを使用して送信された文字列をエンコードします。エンコードが与えられていない場合 UTF-8 エンコードが使用されます。エンコードされた文字列はbase64Binary 文字に変換され、エンコードされたbase64Binary 値が返されます。最初にUTF-8 エンコードがサポートされ、サポートは以下のエンコードに拡張されます: `US-ASCII, ISO-8859-1, UTF-16, UTF-16LE, UTF-16BE, ISO-10646-UCS2, UTF-32, UTF-32LE, UTF-32BE, ISO-10646-UCS4`

サンプル

- **altova:encode-string**("Altova") はUTF-8 エンコード済み文字列 "Altova" の等価である base64Binary を返します。
- **altova:encode-string**("Altova", "UTF-8") はUTF-8 エンコード済み文字列 "Altova" の等価であるbase64Binary のを返します。

get-temp-folder [altova:]

altova:get-temp-folder() を xs:string とする XP2 XQ1 XP3.1 XQ3.1
この関数は引数を必要としません。この関数は現在のユーザーの一時的なフォルダーへのパスを返します。

サンプル

- **altova:get-temp-folder**() は、マシーン上で、xs:string として C:\Users\<UserName>\AppData\Local\Temp\ と類似したパスを返します。

generate-guid [altova:]

altova:generate-guid() asxs:string XP2 XQ1 XP3.1 XQ3.1
ユニークな文字列 GUID 文字列を生成します。

サンプル

- **altova:generate-guid**() は(例えば) 85F971DA-17F3-4E4E-994E-99137873ACCD を返します。

high-res-timer [altova:]

altova:high-res-timer() asxs:double XP3.1 XQ3.1
秒数でシステムハイれぞリューションタイマーの値を返します。ハイれぞリューションタイマーがシステム内で存在すると、必要とされる場合、高度に正確な時間の計算を有効化します(例えば、アニメーションと正確なコード実行の時間の決定などが例として挙げられます)。この関数は、システムのタイマーに精度を与えます。

サンプル

- **altova:high-res-timer**() は '1.16766146154566E6' などを返します。

parse-html [altova:]

altova:parse-html(HTMLText *as xs:string*) **asnode()** XP3.1 XQ3.1
HTMLText 引数は、HTML ドキュメントのテキストを含む文字列です。関数は、文字列からHTML ツリーを作成します。提供された文字列は、HTML 要素を含む、または、含まない場合があります。いずれの場合でも、ツリーのルート要素は、**HTML** と名付けられます。提出された文字列内のHTML コードが有効な HTMLであることを確認することが奨励されます。.

サンプル

- **altova:parse-html**("<html><head/><body><h1>Header</h1></body></html>") は提供された文字列からHTML ツリーを作成します。

(C) 2015-2021 Altova GmbH

▼ sleep[altova:]

altova:sleep(Millisecs *as xs:integer*) **asempty-sequence()** XP2 XQ1 XP3.1 XQ3.1
Millisecs により与えられるミリ秒で示される期間のための実行を延期します。
⊟ *サンプル*

- **altova:sleep**(1000) は1000 ミリ秒のための実行を延期します。

[[トップ] 1559]

26.2.2 その他の拡張関数

Java や C# などのプログラミング言語には、XPath 2.0 / XQuery 関数、または XSLT 2.0 関数として利用できない関数がいくつかあります。そのような関数の良い例として、Java で利用することのできる sin() や cos() といった数学関数があります。XSLT スタイルシートや XQuery のクエリにてこれらの関数が利用できるのであれば、スタイルシートやクエリの適用範囲を大幅に拡張することができ、スタイルシート作成タスクの負担が大幅に軽減されます。Altova 製品で使用されている Altova エンジン(XSLT 1.0、XSLT 2.0、XQuery 1.0)では、Java 1561 や .NET 1569 および MSXSL scripts for XSLT 1575 における拡張関数の使用がサポートされます。 このセクションでは、拡張機能および XSLT スタイルシート内で MSXSL スクリプト。および XQuery ドキュメントを使用する方法について記述します。使用できる拡張関数は以下のように構成されます:

- Java 拡張関数 1561
- .NET 拡張関数 1569
- XSLT に対する MSXSL スクリプト 1575

記述の中では特に、(i) 関連するライブラリ内の関数がどのように呼ばれるか、(ii) 関数呼び出しを行う際に入力として使用される引数を変換するのにどのようなルールが適用され、return により値が返される際にどのような変換ルールが適用されるのか(XSLT/XQuery データオブジェクトに対する関数の結果)について説明されます。

必要条件

拡張関数のサポートを有効にするには、XSLT 変換や XQuery の実行を行うコンピューターに Java Runtime Environment (Java 関数にアクセスする場合)ならびに .NET Framework 2.0 以上(.NET 関数にアクセスする場合)がインストールされている、またはアクセスできる環境が整っている必要があります。

26.2.2.1 Java 拡張関数

Java 拡張関数は、XPath または XQuery 条件式にて使用することができるほか、Java のコンストラクターを呼び出したり、Java の(静的またはインスタンス)メソッドを呼び出すことができます。

Java クラスのフィールドは、引数を持たないメソッドとして扱われます。フィールドは静的またはインスタンスとして存在することができます。フィールドへのアクセス方法については、静的とインスタンスの両方について、以下のサブセクションにて記述されます。

このセクションは以下のサブセクションにより構成されます:

- Java : コンストラクター 1566
- Java : 静的メソッドと静的フィールド 1567

- Java：インスタンスメソッドとインスタンスフィールド 1567
- データ型：XPath/XQuery から Java へ 1568
- データ型：Java から XPath/XQuery へ 1569

以下の点に注意してください：

- Altova デスクトップ製品を使用している場合、Altova アプリケーションは、Java 仮想マシンへのパスを（以下の順序で）読み取ることにより自動的に検知しようとします：(i) Windows レジストリ (ii) **JAVA_HOME** 環境変数。アプリケーションのオプションダイアログ内にカスタムパスを追加することもできます。このエントリは自動的に検知された他の Java VM パス以上の優先順位を有します。
- Altova サーバー製品が Windows マシン上で作動している場合、Java 仮想マシンへのパスは Windows レジストリから最初に読み取られます。成功しない場合、**JAVA_HOME** 環境変数が使用されます。
- Altova サーバー製品を作動する場合、および、Linux または macOS マシン上でサーバー製品を作動する場合、Java 仮想マシンへのパスが **JAVA_HOME** 環境変数内に保管されていることを確認してください。**\bin\server** または **\bin\client** ディレクトリ内の **jvm.dll** ファイルを指している必要があります。

拡張関数のフォーム

XPath/XQuery 条件式における拡張関数では、`prefix:fname()` の形式を取る必要があります。

- `prefix:` 部により拡張関数が Java 関数として認識されます。`java:` から始まる URI のスコープ内の名前空間宣言に拡張関数を関連付けることで Java 関数であるという認識が行われます。名前空間の宣言により、例えば `xmlns:myns="java:java.lang.Math"` という Java クラスが特定されます。名前空間の宣言は、（コロン無しの）`xmlns:myns="java"` という形式で、Java クラスの識別子を拡張関数にある `fname()` 部の左型に配置することでも行うことができます。
- `fname()` 部により、呼び出されている Java メソッドが識別され、メソッドの引数が提供されます（以下の例を参照ください）。`prefix:` 部にて識別された名前空間 URI が Java クラスを識別できなかった場合、Java クラスの識別はクラスの前にくる `fname()` 部にて行うことになり、ピリオドによりクラスから分離されることになります（以下にある2番目の XSLT サンプルを参照）。

メモ 呼び出されるクラスはコンピューターのクラスパス上にある必要がありあす。

XSLT サンプル

以下に静的メソッドを呼び出す2つのサンプルを示します。最初のサンプルでは、クラス名（`java.lang.Math`）が名前空間 URI に加えられており、`fname()` へ加えることはできません。2 番目のサンプルでは、`prefix:` 部に `java:` が与えられており、`fname()` 部にてクラスとメソッドが識別されます。

```
<xsl:value-of xmlns:jMath="java:java.lang.Math"
              select="jMath:cos(3.14)" />

<xsl:value-of xmlns:jmath="java"
              select="jmath:java.lang.Math.cos(3.14)" />
```

拡張関数内にあるメソッド名（上の例では `cos()` ）は、名前付き Java クラス（上の例では `java.lang.Math` ）の public な静的メソッドの名前に一致する必要があります。

XQuery サンプル

以下に XSLT のサンプルに似た XQuery のサンプルを示します：

```
<cosine xmlns:jMath="java:java.lang.Math">
   {jMath:cos(3.14)}
</cosine>
```

ユーザー定義された Java クラス

独自の Java クラスやメソッドを作成した場合、(i) JAR ファイル(または class ファイル)を介してこれらクラスファイルへアクセスしているか、(ii) これら(JAR または class)ファイルが、カレントディレクトリ(XSLT や XQuery ドキュメントが存在するディレクトリ)に配置されているかにより、これらクラスの呼び出し方法が変わってきます。これらファイルの特定方法については、ユーザー定義クラスファイル 1563 ならびにユーザー定義 JAR ファイル 1565 を参照ください。カレントディレクトリには無いクラスファイルや JAR ファイルへのパスは指定しなければならないことご注意してください。

26.2.2.1.1 ユーザー定義のクラスファイル

アクセスがクラスファイルを介したものである場合、4 つのケースが考えられます:

- クラスファイルはパッケージである。XSLT または XQuery ファイルが Java パッケージと同じ場所に収められている。(下のサンプルを参照 1563)
- クラスファイルはパッケージでは無い。XSLT または XQuery ファイルが Java パッケージと同じ場所に収められている。(下のサンプルを参照 1564)
- クラスファイルはパッケージである。XSLT または XQuery ファイルがランダムな場所に収められている。(下のサンプルを参照 1564)
- クラスファイルはパッケージである。XSLT または XQuery ファイルがランダムな場所に収められている。(下のサンプルを参照 1563)

クラスファイルがパッケージではなく、XSLT または XQuery ドキュメントと同じ場所に収められているケースを考えてみましょう。この場合、フォルダー内の全クラスを発見することができるため、ファイルの場所を指定する必要はありません。クラスの識別を行う構文は以下のようになります:

```
java:classname
```

ここで、

`java:` によりユーザー定義の Java 関数が呼ばれていることが示されます(デフォルトでカレントディレクトリにある Java クラスがロードされます)。

`classname` は目的となるメソッドのクラスが含まれているクラスの名前です。

クラスが名前空間 URI にて識別され、名前空間がメソッド呼び出しにて使用されます。

クラスファイルがパッケージで、XSLT/XQuery ファイルが Java パッケージと同じ場所に収められている

以下の例では、`com.altova.extfunc` パッケージにある `Car` クラスの `getVehicleType()` メソッドが呼び出されています。`com.altova.extfunc` パッケージは `JavaProject` という名前のフォルダーに置かれており、XSLT ファイルも同じフォルダーに配置されています。

```
<xsl:stylesheet version="2.0"
       xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
       xmlns:xs="http://www.w3.org/2001/XMLSchema"
       xmlns:fn="http://www.w3.org/2005/xpath-functions"
       xmlns:car="java:com.altova.extfunc.Car" >
<xsl:output exclude-result-prefixes="fn car xsl fo xs"/>

<xsl:template match="/">
```

```
   <a>
   <xsl:value-of select="car:getVehicleType()"/>
   </a>
</xsl:template>

</xsl:stylesheet>
```

クラスファイルが参照され、XSLT/XQuery ファイルがクラスファイルと同じフォルダーに収められている

下のサンプルでは Car クラスの getVehicleType() メソッドが呼び出されます。以下を述べることができます: (i) Car クラスは次のフォルダー内にあります: JavaProject/com/altova/extfunc 。(ii) このフォルダーは下のサンプルの現在のフォルダー内に存在します。XSLT ファイルもフォルダー JavaProject/com/altova/extfunc 内にあります。

```
<xsl:stylesheet version="2.0"
       xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
       xmlns:xs="http://www.w3.org/2001/XMLSchema"
       xmlns:fn="http://www.w3.org/2005/xpath-functions"
       xmlns:car="java:Car" >
<xsl:output exclude-result-prefixes="fn car xsl fo xs"/>

<xsl:template match="/">
   <a>
   <xsl:value-of select="car:getVehicleType()"/>
   </a>
</xsl:template>

</xsl:stylesheet>
```

クラスファイルがパッケージされXSLT/XQuery ファイルがランダムな場所に収められている

以下の例では、com.altova.extfunc パッケージにある Car クラスの getVehicleColor() メソッドが呼び出されています。com.altova.extfunc パッケージは JavaProject という名前のフォルダーに置かれており、XSLT ファイルが任意の場所に配置されています。この場合、以下のような構文でパッケージの場所をクエリ文字列として URI 内にて指定する必要があります:

```
       java:classname[?path=uri-of-package]
```

ここで、

java: によりユーザー定義の Java 関数が呼ばれていることを表します。
uri-of-package は Java パッケージの URI です。
classname は目的のメソッドが含まれているクラス名です。

クラスは名前空間 URI により特定され、名前空間がメソッド呼び出しのプレフィックスで使用されます。以下の例ではカレントディレクトリ以外にあるクラスファイルへのアクセスを行うことができます。

```
<xsl:stylesheet version="2.0"
       xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
       xmlns:xs="http://www.w3.org/2001/XMLSchema"
       xmlns:fn="http://www.w3.org/2005/xpath-functions"
       xmlns:car="java:com.altova.extfunc.Car?path=file:///C:/JavaProject/" >

<xsl:output exclude-result-prefixes="fn car xsl xs"/>

<xsl:template match="/">
   <xsl:variable name="myCar" select="car:new('red')" />
```

```
   <a><xsl:value-of select="car:getCarColor($myCar)"/></a>
</xsl:template>

</xsl:stylesheet>
```

クラスファイルがパッケージではなく、XSLT/XQuery ファイルがランダムな場所に収められている

以下の例では、com.altova.extfunc パッケージにある Car クラスの getCarColor() メソッドが呼び出されています。com.altova.extfunc パッケージは JavaProject という名前のフォルダーに置かれており、XSLT ファイルが任意の場所に配置されています。以下のような構文で、クラスファイルの場所をクエリ文字列として URI 内にて指定する必要があります:

```
java:classname[?path=uri-of-classfile]
```

ここで

java: によりユーザー定義の Java 関数が呼ばれていることを表します。
uri-of-classfile は Java パッケージの URI です。
classname は目的のメソッドが含まれているクラス名です。

クラスは名前空間 URI により特定され、名前空間はメソッド呼び出しのプレフィックスで使用されます。以下の例ではカレントディレクトリ以外にあるクラスファイルへのアクセスを行うことができます。

```
<xsl:stylesheet version="2.0"
       xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
       xmlns:xs="http://www.w3.org/2001/XMLSchema"
       xmlns:fn="http://www.w3.org/2005/xpath-functions"
       xmlns:car="java:Car?path=file:///C:/JavaProject/com/altova/extfunc/" >

<xsl:output exclude-result-prefixes="fn car xsl xs"/>

<xsl:template match="/">
   <xsl:variable name="myCar" select="car:new('red')" />
   <a><xsl:value-of select="car:getCarColor($myCar)"/></a>
</xsl:template>

</xsl:stylesheet>
```

メモ パスが外部関数により与えられている場合、ClassLoader によりパスが追加されます。

26.2.2.1.2 ユーザー定義の JAR ファイル

JAR ファイル経由でアクセスが行われた場合、以下の構文により JAR ファイルの URI を指定する必要があります:

```
xmlns:classNS="java:classname?path=jar:uri-of-jarfile!/"
```

クラスの識別を行う名前空間 URI のプレフィックスを使用してメソッドが呼び出されます: classNS:method()

上の例に対する説明は以下のとおりです:

java: 関数が呼び出されていることを表します。
classname がユーザー定義されたクラスの名前になります。
? はクラス名とパスを分離するために使用されます。
path=jar: により、JAR ファイルへのパスが与えられていることを示します。
uri-of-jarfile は JAR ファイルの URI となります。

`!/` は、終了を表すデリミタとなります。
`classNS:method()` により、メソッドの呼び出しが行われます。

その他にも、メソッド名とともにクラス名を与えることができます。構文の例を以下に示します:

```
            xmlns:ns1="java:docx.layout.pages?
path=jar:file:///c:/projects/docs/docx.jar!/"
            ns1:main()

            xmlns:ns2="java?path=jar:file:///c:/projects/docs/docx.jar!/"
            ns2:docx.layout.pages.main()
```

以下に JAR ファイルを使った Java 拡張関数を呼び出す XSLT サンプルを記します:

```
<xsl:stylesheet version="2.0"
      xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
      xmlns:xs="http://www.w3.org/2001/XMLSchema"
      xmlns:fn="http://www.w3.org/2005/xpath-functions"
      xmlns:car="java?path=jar:file:///C:/test/Car1.jar!/" >
<xsl:output exclude-result-prefixes="fn car xsl xs"/>

<xsl:template match="/">
      <xsl:variable name="myCar" select="car:Car1.new('red')" />
      <a><xsl:value-of select="car:Car1.getCarColor($myCar)"/></a>
</xsl:template>

<xsl:template match="car"/>

</xsl:stylesheet>
```

メモ: 拡張関数によりパスが与えられている場合、ClassLoader にパスが追加されます。

26.2.2.1.3 Java:コンストラクター

拡張関数を使用することで Java コンストラクターを呼び出すことができます。`new()` により全てのコンストラクターを呼び出すことができます。

Java コンストラクターの呼び出し結果を、黙示的に XPath/XQuery データ型へ変換 1569 できる場合、Java 拡張関数により XPath/XQuery データ型のシーケンスが返されます。Java コンストラクターの呼び出し結果が XPath/XQuery データ型へ変換できない場合、値を返すクラス名でラップした Java オブジェクトがコンストラクターにより作成されます。例えば、`java.util.Date` クラスに対するコンストラクターが呼び出された場合 (`java.util.Date.new()`)、`java.util.Date` を持ったオブジェクトが返されます。返されたオブジェクトのレキシカルフォーマットは、XPath データ型のレキシカルフォーマットにマッチしない場合もあり、目的の XPath データ型に対するレキシカルフォーマットへ値の変換を行い、その後目的の XPath データ型へ変換を行う必要があります。

コンストラクターにより作成された Java オブジェクトにより2つのことが行えます:

- 変数への割り当てを行うことができます:
  ```
  <xsl:variable name="currentdate" select="date:new()"
  xmlns:date="java:java.util.Date" />
  ```
- 拡張関数への受け渡しを行うことができます (*インスタンスメソッドならびにインスタンスフィールド* 1567 を参照ください):
  ```
  <xsl:value-of select="date:toString(date:new())"
  xmlns:date="java:java.util.Date" />
  ```

 (C) 2015-2021 Altova GmbH

26.2.2.1.4 Java：静的メソッドと静的フィールド

静的メソッドは、Java 名ならびにメソッドの引数により直接呼び出すことができます。E や PI といった定数の静的フィールド（引数を持たないメソッド）は、引数を指定することなくアクセスすることができます。

XSLT の例

静的メソッドならびにフィールドを呼び出す例を以下に示します:

```
<xsl:value-of xmlns:jMath="java:java.lang.Math"
              select="jMath:cos(3.14)" />

<xsl:value-of xmlns:jMath="java:java.lang.Math"
              select="jMath:cos( jMath:PI() )" />

<xsl:value-of xmlns:jMath="java:java.lang.Math"
              select="jMath:E() * jMath:cos(3.14)" />
```

上の拡張関数は prefix:fname() という形式を使用していることに注意してください。3つの例にあるプレフィックスは全て jMath: となっており、このプレフィックスは、java:java.lang.Math という名前空間 URI に関連付けられています。名前空間 URI は java: で開始しなければなりません。上の例では、クラス名（java.lang.Math）を含むように拡張されています。拡張関数の fname() 部は（java.lang.Math のような）public クラスにマッチする必要があり、その後 public な静的メソッドが引数とともに続くか（例: cos(3.14)）、public な静的フィールドが続きます（例: PI()）。

上の例では、クラス名が名前空間 URI に含まれています。クラス名が名前空間 URI に含まれていない場合、以下の例にあるように、拡張関数の fname() 部にて追加する必要があります:

```
<xsl:value-of xmlns:java="java:"
              select="java:java.lang.Math.cos(3.14)" />
```

XQuery の例

XQuery における似たようなサンプルを以下に示します:

```
<cosine xmlns:jMath="java:java.lang.Math">
   {jMath:cos(3.14)}
</cosine>
```

26.2.2.1.5 Java：インスタンスメソッドとインスタンスフィールド

メソッド呼び出しの第一引数として パースされる Java オブジェクトが、インスタンスメソッドには与えられています。このような Java オブジェクトは通常、拡張関数を使うことで作成される（例: コンストラクターの呼び出し）か、スタイルシート パラメーター/変数により作成されます。以下に XSLT サンプルを示します:

```
<xsl:stylesheet version="1.0" exclude-result-prefixes="date"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:date="java:java.util.Date"
  xmlns:jlang="java:java.lang">
  <xsl:param name="CurrentDate" select="date:new()"/>
  <xsl:template match="/">
```

```
      <enrollment institution-id="Altova School"
                date="{date:toString($CurrentDate)}"
                type="{jlang:Object.toString(jlang:Object.getClass( date:new() ))}">
      </enrollment>
  </xsl:template>
</xsl:stylesheet>
```

上の例では、ノード enrollment/@type の値が以下のように作成されます:

1. java.util.Date クラスに対するオブジェクトがコンストラクター(date:new() コンストラクター)とともに作成されます。
2. jlang.Object.getClass メソッドの引数として Java オブジェクトがパースされます。
3. getClass メソッドにより得られたオブジェクトが jlang.Object.toString メソッドの引数としてパースされます。

結果(@type の値)は java.util.Date を持った文字列となります。

インスタンスフィールドは、引数としてインスタンスフィールドへ渡される Java オブジェクトではないという点で、理論的にはインスタンスメソッドと異なります。パラメーターや変数がその代わりに引数として渡されますが、パラメーター/変数そのものに Java オブジェクトから返された値が含まれている場合もあります。例えば、CurrentDate パラメーターには java.util.Date クラスのコンストラクターから返された値が含まれます。この値は引数として、date:toString インスタンスメソッドへ渡され、/enrollment/@date の値として使用されます。

26.2.2.1.6 データ型:XPath/XQuery から Java へ

XPath/XQuery 条件式内部から Java 関数が呼び出された場合、複数ある同名の Java クラスのうち、どのクラスが呼び出されたのか決定するのに、関数へ渡される引数のデータ型が重要になります。

Java では、以下のルールが適用されます:

- 同名の Java メソッドが2つ以上あり、それぞれが違う数の引数を受け取る場合、呼び出しに使用されている引数の数に一番マッチするメソッドが選択されます。
- XPath/XQuery の文字列、数値、boolean データ型は、黙示的に対応する Java データ型へ変換されます(以下のリストを参照)。与えられた XPath/XQuery 型が2つ以上の Java 型へ変換できる場合(例: xs:integer)、選択されたメソッドにて宣言されている Java 型が使用されます。例えば、呼び出された Java メソッドが fx(decimal) で、与えられた XPath/XQuery データ型が xs:integer の場合、xs:integer が Java の decimal データ型へ変換されます。

以下のテーブルに、XPath/XQuery の文字列、数値、boolean 型から Java データ型への黙示的な変換リストを示します。

xs:string	java.lang.String
xs:boolean	boolean (プリミティブ型), java.lang.Boolean
xs:integer	int, long, short, byte, float, double, ならびに、java.lang.Integer のようなこれらのラッパークラス
xs:float	float (プリミティブ型), java.lang.Float, double (プリミティブ型)
xs:double	double (プリミティブ型), java.lang.Double
xs:decimal	float (プリミティブ型), java.lang.Float, double(プリミティブ型), java.lang.Double

(C) 2015-2021 Altova GmbH

上のリストにある XML スキーマデータ型(ならびに XPath や XQuery で使用されているデータ型)のサブタイプも、対応する祖先のサブタイプとして Java のデータ型へ変換されます。

場合によっては、与えられた情報から正しい Java メソッドを選択することができない場合もあります。例えば、以下のような場合を考えてみましょう:

- 与えられた引数が 10 という値を持った xs:untypedAtomic 型で、`mymethod(float)` メソッドへ渡されるのを意図している。
- しかし、そのクラスには別のデータ型を取る `mymethod(double)` というメソッドも存在する。
- メソッド名が同じで、与えられた型(`xs:untypedAtomic`)も `float` と `double` の両方に変換することができるため、`xs:untypedAtomic` が `float` ではなく `double` に変換される可能性もある。
- 結果として、意図したメソッドは選択されず、予期しない動作結果がを招く可能性がある。この問題を回避するには、意図したメソッドを使用するユーザー定義のメソッドを別の名前で新たに作成する必要があります。

上のリストでカバーされていない型(例: `xs:date`)は変換されず、エラーとなります。しかし場合によっては、Java コンストラクターを使用して、目的の Java データ型を作成することが可能だということも留意してください。

26.2.2.1.7 データ型:Java から XPath/XQuery へ

Java メソッドにより値が返され、値のデータ型が文字列、数値、または boolean 型の場合、対応する XPath/XQuery 型への変換が行われます。例えば、Java の `java.lang.Boolean` や `boolean` データ型は `xsd:boolean` へ変換されます。

関数から返された一次元配列は、シーケンス(sequence)に展開されます。2 次元以上の配列は変換されることが無いため、ラップして使用するべきでしょう。

Java オブジェクトや文字列、数値、boolean 以外のデータ型がラップされて返された場合、最初に(例えば `toString` といった)Java メソッドを使用して Java オブジェクトを文字列へ変換することで、目的の XPath/XQuery 型への変換を行います。XPath/XQuery では、文字列を目的となる型のレキシカルフォーマットへ変換し、目的の型への変換を(例えば `cast as` 式を使用することで)行うことができます。

26.2.2.2 .NET 拡張関数

.NET プラットフォームにて作業を行なっている場合、.NET 言語(例えば C#)で記述された拡張関数を使用することができます。.NET 拡張関数は XPath や XQuery 条件式内部から使用することができ、.NET クラス内部にあるコンストラクターや(static または インスタンス変数)プロパティを呼び出すことができます。

`get_PropertyName()` 構文を使用することにより .NET クラスのプロパティを呼び出すことができます。

このセクションは、以下のサブセクションにより構成されています:

- .NET コンストラクター 1571
- .NET:静的メソッドならびに静的フィールド 1572
- .NET:インスタンスメソッドとインスタンスフィールド 1573
- データ型:XPath/XQuery から .NET へ 1574
- データ型:.NET から XPath/XQuery へ 1574

拡張関数のフォーム

XPath/XQuery 条件式にある拡張関数は、`prefix:fname()` の形式を取る必要があります。

- `prefix:` 部は、呼び出されている.NET クラスを特定するURI となります。
- `fname()` 部により、.NET クラス内にあるコンストラクター、プロパティ、または(静的またはインスタンス)メソッドが特定され、必要な場合は引数が与えられます。
- URI は`clitype:` で開始する必要があり、これにより関数が.NET 拡張関数であることが認識されます。
- 拡張関数の `prefix:fname()` 形式は、システムクラスならびにロードされたアセンブリとともに使用することもできます。しかしクラスをロードする必要がある場合、必要な情報が含まれるパラメーターが必要になります。

パラメーター

アセンブリをロードするには以下のパラメーターを使用してください:

`asm`	ロードするアセンブリの名前。
`ver`	バージョン番号(ピリオドにより分離された最大4桁の整数)。
`sn`	アセンブリ厳密名のキートーケン(16新数の数値)。
`from`	ロードするアセンブリ(DLL) の場所を特定するURI。URI が相対パスの場合、XSLT やXQuery ドキュメントに対して相対的になります。このパラメーターが指定された場合、その他のパラメーターが無視されます。
`partialname`	アセンブリ名の一部。`Assembly.LoadWith.PartialName()` へ渡され、アセンブリへのロードが試みられます。 `partialname` が指定された場合、その他のパラメーターが無視されます。
`loc`	例えば`en-US` というロケール。デフォルトは`neutral` です。

アセンブリがDLL からロードされる場合、`from` パラメータが使用して、`sn` パラメーターは使用しないでください。アセンブリがグローバルアセンブリキャッシュ(GAC) からロードされる場合、`sn` パラメータを使用して `from` パラメーターは使用しないでください。

最初のパラメーターの前に疑問符 (?) を挿入し、パラメーター同士はセミコロンで分離する必要があります。パラメーター名へ値を受け渡すには、統合符号 (=) を使用します(以下の例を参照ください)。

名前空間宣言の例

XSLT において、システムクラス `System.Environment` を特定する名前空間宣言の例を以下に示します:

```
xmlns:myns="clitype:System.Environment"
```

XSLT において、ロードするクラスを `Trade.Forward.Scrip` として特定する名前空間宣言の例を以下に示します:

```
xmlns:myns="clitype:Trade.Forward.Scrip?asm=forward;version=10.6.2.1"
```

XQuery において、システムクラス `MyManagedDLL.testClass` を特定する名前空間宣言の例を以下に示します。2つのケースが考えられます:

1. アセンブリがGAC からロードされた場合:
```
declare namespace cs="clitype:MyManagedDLL.testClass?asm=MyManagedDLL;
       ver=1.2.3.4;loc=neutral;sn=b9f091b72dccfba8";
```


(C) 2015-2021 Altova GmbH

2. アセンブリがDLL からロードされた場合(完全参照と一部の参照):

```
declare namespace cs="clitype:MyManagedDLL.testClass?from=file:///C:/Altova
Projects/extFunctions/MyManagedDLL.dll;

declare namespace cs="clitype:MyManagedDLL.testClass?from=MyManagedDLL.dll;
```

XSLT の例

システムクラス System.Math 内の関数を呼び出すための完全な XSLT の例を以下に示します:

```
<xsl:stylesheet version="2.0"
   xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
   xmlns:xs="http://www.w3.org/2001/XMLSchema"
   xmlns:fn="http://www.w3.org/2005/xpath-functions">
   <xsl:output method="xml" omit-xml-declaration="yes" />
   <xsl:template match="/">
      <math xmlns:math="clitype:System.Math">
         <sqrt><xsl:value-of select="math:Sqrt(9)"/></sqrt>
         <pi><xsl:value-of select="math:PI()"/></pi>
         <e><xsl:value-of select="math:E()"/></e>
         <pow><xsl:value-of select="math:Pow(math:PI(), math:E())"/></pow>
      </math>
   </xsl:template>
</xsl:stylesheet>
```

math 要素にある名前空間宣言により、math: プレフィックスと clitype:System.Math URI が関連付けられます。URI の最初にある clitype: により、それ以降の記述がシステムクラスまたはロードされたクラスを特定するものであることが示されます。XPath 条件式にある math: プレフィックスにより、拡張関数が URI (そしてクラス) System.Math に関連付けられます。拡張関数により、System.Math クラス内のメソッドが特定され、必要な場所に引数が与えられます。

XQuery の例

上の XSLT に対する例と同様の XQuery 例を以下に示します:

```
<math xmlns:math="clitype:System.Math">
   {math:Sqrt(9)}
</math>
```

上の XSLT と同様に、名前空間宣言により .NET クラス(この場合はシステムクラス) が特定されます。XQuery 式により呼び出されるメソッドが特定され、引数が与えられます。

26.2.2.2.1 .NET コンストラクター

拡張関数を使用することで、.NET コンストラクターを呼び出すことができます。new() により全てのコンストラクターを呼び出すことができます。クラス内に2つ以上のコンストラクターがある場合、与えられた引数の数が最もマッチするコンストラクターが選択されます。与えられた引数に対してマッチするコンストラクターが見つからない場合、"No constructor found'" エラーが返されます。

XPath/XQuery データ型を返すコンストラクター

.NET コンストラクター呼び出しの結果が XPath/XQuery データ型へ黙示的に変換[1569]することができる場合、.NET 拡張関数から XPath/XQuery データ型のシーケンスが返されます。

.NET オブジェクトを返すコンストラクター

.NET コンストラクター呼び出しの結果がXPath/XQuery データ型へ適切に変換できない場合、値を返すクラス名でラップした.NET オブジェクトがコンストラクターにより作成されます。例えば、`System.DateTime` クラスのコンストラクターが（`System.DateTime.new()` により）呼ばれた場合 `System.DateTime` 型を持つオブジェクトが返されます。

返されたオブジェクトのレキシカルフォーマットは、目的のXPath データ型と違っている場合があります。その場合、返された値を: (i) 目的の XPath データ型のレキシカルフォーマットへ変換し、(ii) 目的のXPath データ型へキャストする必要があります。

コンストラクターにより作成された.NET オブジェクトに対して3つのことを行うことができます:

- 変数内で使用することができます:
```
<xsl:variable name="currentdate" select="date:new(2008, 4, 29)"
xmlns:date="clitype:System.DateTime" />
```
- 拡張関数へ渡すことができます（*インスタンスメソッドとインスタンスフィールド*[1567] を参照ください）:
```
<xsl:value-of select="date:ToString(date:new(2008, 4, 29))"
xmlns:date="clitype:System.DateTime" />
```
- 文字列、数値、または boolean へ変換することができます:
```
<xsl:value-of select="xs:integer(date:get_Month(date:new(2008, 4, 29)))"
xmlns:date="clitype:System.DateTime" />
```

26.2.2.2.2 .NET：静的メソッドと静的フィールド

メソッド名と引数を与えることで、静的メソッドを直接呼び出すことができます。呼び出しに使用される名前は、クラス内にある public static メソッドと完全に一致する必要があります。関数の呼び出しに使用されたメソッド名と引数の数にマッチするものがクラス内に複数ある場合、与えられた引数が評価され、最もマッチするものが選択されます。マッチする結果が得られない場合、エラーが返されます。

メモ: .NET クラス内にあるフィールドは、引数を持たないメソッドとしてみなされます。プロパティは `get_PropertyName()` 構文により呼び出されます。

例

1つの引数とともにメソッド（`System.Math.Sin(arg)`）を呼び出す XSLT サンプルを以下に示します:

```
<xsl:value-of select="math:Sin(30)" xmlns:math="clitype:System.Math"/>
```

（引数なしのメソッドとしてみなされる）フィールド（`System.Double.MaxValue()`）を呼び出す XSLT サンプルを以下に示します:

```
<xsl:value-of select="double:MaxValue()" xmlns:double="clitype:System.Double"/>
```

（`get_PropertyName()` 構文を使って）プロパティ（`System.String()`）を呼び出す XSLT サンプルを以下に示します:

```
<xsl:value-of select="string:get_Length('my string')"
xmlns:string="clitype:System.String"/>
```

1つの引数とともにメソッド（(System.Math.Sin(arg)）を呼び出す XQuery サンプルを以下に示します:

```
<sin xmlns:math="clitype:System.Math">
   { math:Sin(30) }
</sin>
```

26.2.2.2.3 .NET：インスタンスメソッドとインスタンスフィールド

インスタンスメソッドには、メソッド呼び出しの第一引数として.NET オブジェクトが渡されます。通常この.NET オブジェクトは、拡張関数（例えばコンストラクター呼び出し）またはスタイルシート パラメーター/変数により作成されます。以下にXSLT の例を示します:

```
<xsl:stylesheet version="2.0"
   xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
   xmlns:xs="http://www.w3.org/2001/XMLSchema"
   xmlns:fn="http://www.w3.org/2005/xpath-functions">
   <xsl:output method="xml" omit-xml-declaration="yes"/>
   <xsl:template match="/">
      <xsl:variable name="releasedate"
         select="date:new(2008, 4, 29)"
         xmlns:date="clitype:System.DateTime"/>
      <doc>
         <date>
            <xsl:value-of select="date:ToString(date:new(2008, 4, 29))"
               xmlns:date="clitype:System.DateTime"/>
         </date>
         <date>
            <xsl:value-of select="date:ToString($releasedate)"
               xmlns:date="clitype:System.DateTime"/>
         </date>
      </doc>
   </xsl:template>
</xsl:stylesheet>
```

上の例では、System.DateTime コンストラクター（new(2008, 4, 29)）が、System.DateTime 型の.NET オブジェクトの作成に使用されます。このオブジェクトは、最初にreleasedate 変数の値として、次にSystem.DateTime.ToString() メソッドの引数として作成されます。System.DateTime.ToString() インスタンスメソッドは、System.DateTime コンストラクターの（new(2008, 4, 29)）における引数として2度呼び出されます。これらインスタンスにおいて releasedate 変数が.NET オブジェクトを取得するのに使用されます。

インスタンスメソッドとインスタンスフィールド

インスタンスメソッドとインスタンスフィールドの違いは理論的なものです。インスタンスメソッドでは.NET オブジェクトが直接引数に渡され、インスタンスフィールドではパラメーターや変数が（.NET オブジェクトそのものを含めることはできるものの）代わりに渡されます。例えば上の例では、releasedate 変数に.NET オブジェクトが含まれており、この変数が2番目のdate 要素コンストラクターにて ToString() の引数として渡されます。そのため、最初のdate 要素にある ToString() インスタンスがインスタンスメソッドであるのに対し、2番目はインスタンスフィールドとしてみなされます。両方のインスタンスで求められる結果は等価です。

26.2.2.2.4 データ型：XPath/XQuery から .NET へ

.NET 拡張関数がXPath/XQuery 条件式内部で使用された場合、複数ある.NET メソッドのうち、どれが呼び出されたのが決定するのに関数の引数に使用されるデータ型が重要になります。

.NET では、以下のルールが適用されます:

- クラス内に同名のメソッドが2つ以上ある場合、呼び出しに使用された引数の数がマッチするメソッドだけが、呼び出される関数の候補に狭められます。
- XPath/XQuery の文字列、数値、boolean データ型は黙示的に対応する.NET データ型へ変換されます（以下のリストを参照）。与えられたXPath/XQuery 型が2つ以上の.NET 型へ変換できる場合（例: `xs:integer` ）、選択されたメソッドにて宣言されている.NET 型が使用されます。例えば、呼び出された.NET メソッドがfx(fouble) で、与えられたXPath/XQuery データ型がxs:integer の場合、`xs:integer` が.NET の`double` データ型へ変換されます。

以下のテーブルに、XPath/XQuery の文字列、数値、boolean 型から.NET データ型へ行われる黙示的な変換リストを示します。

xs:string	StringValue, string
xs:boolean	BooleanValue, bool
xs:integer	IntegerValue, decimal, long, integer, short, byte, double, float
xs:float	FloatValue, float, double
xs:double	DoubleValue, double
xs:decimal	DecimalValue, decimal, double, float

上のリストにあるXML スキーマ型（ならびにXPath やXQuery で使用されているデータ型）のサブタイプも、対応する祖先のサブタイプとして.NET のデータ型へ変換されます。

場合によっては、与えられた情報から正しい.NET メソッドを選択することができない場合もあります。例えば、以下のような場合を考えてみましょう:

- 与えられた引数が1つある`xs:untypedAtomic` 値で、`mymethod(float)` メソッドへ渡されるのを意図している。
- しかし、そのクラスには別のデータ型をとる`mymethod(double)` というメソッドも存在する。
- メソッド名が同じで、与えられた型（`xs:untypedAtomic`）も`float` と`double` の両方に変換することができるため、`xs:untypedAtomic` が`float` ではなく`double` に変換される可能性もある。
- 結果として、意図したメソッドは選択されず、予期しない動作結果がを招く可能性がある。この問題を回避するには、意図したメソッドを使用するユーザー定義のメソッドを別の名前で新たに作成する必要があります。

上のリストでカバーされていない型（例: `xs:date`）は変換されずエラーとなります。

26.2.2.2.5 データ型：.NET から XPath/XQuery へ

.NET メソッドにより値が返される際に値のデータ型が文字列、数値、またはboolean 型の場合、対応するXPath/XQuery 型への変換が行われます。例えば、.NET の`decimal`データ型は`xsd:decimal` へ変換されます。

(C) 2015-2021 Altova GmbH

.NET オブジェクトや文字列、数値、boolean 以外のデータ型が返された場合、最初に(例えば System.DateTime.ToString() といった).NET メソッドを使用して.NET オブジェクトを文字列へ変換します。XPath/XQuery では、文字列を目的となる型のレキシカルフォーマットへ変換し、目的の型への変換を(例えば cast as 式を使用することで)行うことができます。

26.2.2.3 XSLT に対する MSXSL スクリプト

<msxsl:script> 要素にはユーザー定義の関数や変数が含まれており、XSLT スタイルシート内の XPath 条件式内部から呼び出しを行うことができます。<msxsl:script> はトップレベル要素で、<xsl:stylesheet> または <xsl:transform> の子要素である必要があります。

<msxsl:script> 要素は urn:schemas-microsoft-com:xslt 名前空間内に存在する必要があります(以下を参照ください)。

スクリプト言語と名前空間

ブロック内で使用されるスクリプト言語は <msxsl:script> 要素の language 属性にて指定され、XPath 条件式における関数の呼び出しに対して使用される名前空間は implements-prefix 属性により特定されます(以下を参照)。

```
<msxsl:script language="scripting-language" implements-prefix="user-namespace-prefix">

   function-1 or variable-1
   ...
   function-n or variable-n

</msxsl:script>
```

<msxsl:script> 要素は Windows Scripting Runtime を使ったやりとりを行うため、お使いのコンピューターにインストールされた言語だけしか <msxsl:script> 要素では使用することができません。**MSXSL スクリプトを使用するには.NET Framework 2.0 以上のプラットフォームをインストールする必要があります。**結果として <msxsl:script> 言語から.NET スクリプト言語を使用することができます。

HTML の <script> 要素における language 属性と同じ値が language 属性では受理されます。language 属性が指定されていない場合、Microsoft JScript がデフォルトとして想定されます。

implements-prefix 属性には名前空間スコープ内で宣言されたプレフィックスが与えられます。通常この名前空間は関数ライブラリのために予約されたユーザーの名前空間となります。<msxsl:script>要素内で定義された全ての関数ならびに変数は、implements-prefix 属性にて指定されたプレフィックスで特定される名前空間に収められます。XPath 条件式内部から関数が呼ばれる場合、完全修飾関数名が同じ名前空間内に関数として定義されていなければなりません。

サンプル

<msxsl:script> 要素内で定義された関数を使用する XSLT スタイルシートの例を以下に示します:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="2.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
   xmlns:xs="http://www.w3.org/2001/XMLSchema"
   xmlns:fn="http://www.w3.org/2005/xpath-functions"
   xmlns:msxsl="urn:schemas-microsoft-com:xslt"
   xmlns:user="http://mycompany.com/mynamespace">

  <msxsl:script language="VBScript" implements-prefix="user">
    <![CDATA[
    ' Input:A currency value: the wholesale price
    ' Returns:The retail price: the input value plus 20% margin,
```

```
    ' rounded to the nearest cent
    dim a as integer  = 13
    Function AddMargin(WholesalePrice) as integer
      AddMargin = WholesalePrice * 1.2 + a
    End Function
  ]]>
  </msxsl:script>

  <xsl:template match="/">
    <html>
      <body>
        <p>
          <b>Total Retail Price =
            $<xsl:value-of select="user:AddMargin(50)"/>
          </b>
          <br/>
          <b>Total Wholesale Price =
            $<xsl:value-of select="50"/>
          </b>
        </p>
      </body>
    </html>
  </xsl:template>
</xsl:stylesheet>
```

データ型

スクリプトブロックとのやりとりに使用されるパラメーターの値はXPath データ型に限定されます。スクリプトブロック内にある関数にてやりとりされるデータや変数に、この制限はありません。

アセンブリ

msxsl:assembly 要素を使用することで、アセンブリをスクリプト内部へインポートすることができます。アセンブリは名前やURI により特定されます。アセンブリのインポートは、コンパイル時に行われます。以下に msxsl:assembly 要素の簡単な使用例を示します:

```
<msxsl:script>
       <msxsl:assembly name="myAssembly.assemblyName" />
       <msxsl:assembly href="pathToAssembly" />

       ...

</msxsl:script>
```

アセンブリ名は、以下のような完全な名前でも::

```
"system.Math, Version=3.1.4500.1 Culture=neutral PublicKeyToken=a46b3f648229c514"
```

`"myAssembly.Draw"` のような短い名前でも指定できます。

名前空間

msxsl:using 要素により名前空間の宣言を行うことができます。これにより、スクリプト内において名前空間無しでアセンブリクラスを使用することができ、タイピングの手間を軽減することができます。以下に `msxsl:using` 要素の簡単な使用例を示します:

```
<msxsl:script>
       <msxsl:using namespace="myAssemblyNS.NamespaceName" />

       ...
```

```
</msxsl:script>
```

`namespace` 属性の値は名前空間の名前となります。

26.3 DB 内で生成された XML スキーマのデータ型

XML スキーマがデータベース(DB)から生成されると、そのデータベース特有のデータ型がXML スキーマデータ型に変換されます。一般に使用されるDB データ型からXML スキーマデータ型へのマッピングは付録に記載されています。以下のリストから選択してください。

- ADO 1578
- MS Access 1579
- MS SQL Server 1579
- MySQL 1580
- ODBC 1581
- Oracle 1582
- Sybase 1582

26.3.1 ADO

XML スキーマがADO データベース(DB) から生成される場合、ADO DB データ型は下のテーブルの通りXML スキーマデータ型に変換されます。

ADO データ型	XML スキーマ データ型
adGUID	xs:ID
adChar	xs:string
adWChar	xs:string
adVarChar	xs:string
adWVarChar	xs:string
adLongVarChar	xs:string
adWLongVarChar	xs:string
adVarWChar	xs:string
adBoolean	xs:boolean
adSingle	xs:float
adDouble	xs:double
adNumeric	xs:decimal
adCurrency	xs:decimal
adDBTimeStamp	xs:dateTime
adDate	xs:date
adBinary	xs:base64Binary
adVarBinary	xs:base64Binary
adLongVarBinary	xs:base64Binary
adInteger	xs:Integer

(C) 2015-2021 Altova GmbH

adUnsignedInt	xs:unsignedInt
adSmallInt	xs:short
adUnsignedSmallInt	xs:unsignedShort
adBigInt	xs:long
adUnsignedBigInt	xs:unsignedLong
adTinyInt	xs:byte
adUnsignedTinyInt	xs:unsignedByte

26.3.2 MS Access

XML スキーマがMS Access データベース(DB) から生成される場合、MS Access DB データ型は下のテーブルの通りXML スキーマデータ型に変換されます。

MS Access データ型	XML スキーマ データ型
GUID	xs:ID
char	xs:string
varchar	xs:string
memo	xs:string
bit	xs:boolean
Number(single)	xs:float
Number(double)	xs:double
Decimal	xs:decimal
Currency	xs:decimal
Date/Time	xs:dateTime
Number(Long Integer)	xs:integer
Number(Integer)	xs:short
Number(Byte)	xs:byte
OLE Object	xs:base64Binary

26.3.3 MS SQL Server

XML スキーマがMS SQL Server データベース(DB) から生成される場合、MS SQL Server DB データ型は下のテーブルの通りXML スキーマデータ型に変換されます。

MS SQL Server データ型	XML スキーマ データ型
uniqueidentifier	xs:ID

char	xs:string
nchar	xs:string
varchar	xs:string
nvarchar	xs:string
text	xs:string
ntext	xs:string
sysname	xs:string
bit	xs:boolean
real	xs:float
float	xs:double
decimal	xs:decimal
money	xs:decimal
smallmoney	xs:decimal
datetime	xs:dateTime
smalldatetime	xs:dateTime
binary	xs:base64Binary
varbinary	xs:base64Binary
image	xs:base64Binary
integer	xs:integer
smallint	xs:short
bigint	xs:long
tinyint	xs:byte

26.3.4 MySQL

XML スキーマがMySQL データベース(DB) から生成される場合、MySQL DB データ型は下のテーブルの通りXML スキーマデータ型に変換されます。

MySQL データ型	XML スキーマ データ型
char	xs:string
varchar	xs:string
text	xs:string
tinytext	xs:string
mediumtext	xs:string
longtext	xs:string

tinyint(1)	xs:boolean
float	xs:float
double	xs:double
decimal	xs:decimal
datetime	xs:dateTime
blob	xs:base64Binary
tinyblob	xs:base64Binary
mediumblob	xs:base64Binary
longblob	xs:base64Binary
smallint	xs:short
bigint	xs:long
tinyint	xs:byte

26.3.5 ODBC

XML スキーマが ODBC データベース(DB) から生成される場合、ODBC DB データ型は下のテーブルの通り XML スキーマデータ型に変換されます。

ODBC データ型	XML スキーマ データ型
SQL_GUID	xs:ID
SQL_CHAR	xs:string
SQL_VARCHAR	xs:string
SQL_LONGVARCHAR	xs:string
SQL_BIT	xs:boolean
SQL_REAL	xs:float
SQL_DOUBLE	xs:double
SQL_DECIMAL	xs:decimal
SQL_TIMESTAMP	xs:dateTime
SQL_DATE	xs:date
SQL_BINARY	xs:base64Binary
SQL_VARBINARY	xs:base64Binary
SQL_LONGVARBINARY	xs:base64Binary
SQL_INTEGER	xs:integer
SQL_SMALLINT	xs:short
SQL_BIGINT	xs:long

SQL_TINYINT	xs:byte

26.3.6 Oracle

XML スキーマがOracle データベース(DB) から生成される場合、Oracle DB データ型は下のテーブルの通りXML スキーマデータ型に変換されます。

Oracle データ型	XML スキーマ データ型
ROWID	xs:ID
CHAR	xs:string
NCHAR	xs:string
VARCHAR2	xs:string
NVARCHAR2	xs:string
CLOB	xs:string
NCLOB	xs:string
NUMBER (with check constraint applied)*	xs:boolean
NUMBER	xs:decimal
FLOAT	xs:double
DATE	xs:dateTime
INTERVAL YEAR TO MONTH	xs:gYearMonth
BLOB	xs:base64Binary

* データ型 NUMBER の列にチェック制約が適用されている場合、またチェック制約が0 または1 の値をチェックする場合、この列の NUMBER データ型は、 xs:boolean のXML スキーマデータ型に変換されます。このメカニズムは、生成されたXML スキーマ内にxs:boolean データ型を生成する際に役に立ちます。

26.3.7 Sybase

XML スキーマがSybase データベース(DB) から生成される場合、Sybase DB データ型は下のテーブルの通りXML スキーマデータ型に変換されます。

Sybase データ型	XML スキーマ データ型
char	xs:string
nchar	xs:string
varchar	xs:string
nvarchar	xs:string
text	xs:string

(C) 2015-2021 Altova GmbH

sysname-varchar(30)	xs:string
bit	xs:boolean
real	xs:float
float	xs:float
double	xs:double
decimal	xs:decimal
money	xs:decimal
smallmoney	xs:decimal
datetime	xs:dateTime
smalldatetime	xs:dateTime
timestamp	xs:dateTime
binary<=255	xs:base64Binary
varbinary<=255	xs:base64Binary
image	xs:base64Binary
integer	xs:integer
smallint	xs:short
tinyint	xs:byte

26.4 XML スキーマから生成された DB のデータ型

データベース構造がXML スキーマがから生成されると、そのデータベース特有のデータ型がXML スキーマデータ型から生成されます。一般に使用されるXML スキーマデータ型からDB データ型へのマッピングは付録に記載されています。以下のリストから選択してください。

- MS Access 1584
- MS SQL Server 1585
- MySQL 1587
- Oracle 1589

26.4.1 MS Access

MS Access データベース(DB) がXML スキーマから生成される場合、MS Access データ型は下のテーブルの通りXML スキーマデータ型に変換されます。

XML スキーマ データ型	MS Access データ型
xs:ID	GUID
xs:string	If no facets varchar (255)
	Size = either length or maxLength
	If Size <= 255 varchar (n)
	else memo
xs:normalizedString	Same as xs:string
xs:token	Same as xs:string
xs:Name	Same as xs:string
xs:NCName	Same as xs:string
xs:anyURI	Same as xs:string
xs:QName	Same as xs:string
xs:NOTATION	Same as xs:string
xs:boolean	bit
xs:float	Number (single)
xs:double	Number (double)
xs:decimal	Decimal
xs:duration	Date/Time
xs:dateTime	Date/Time
xs:time	Date/Time
xs:date	Date/Time
xs:gYearMonth	Date/Time

(C) 2015-2021 Altova GmbH

xs:gYear	Date/Time
xs:gMonthDay	Date/Time
xs:gDay	Date/Time
xs:gMonth	Date/Time
xs:hexBinary	If no facets varbinary (255)
	Size = either length or maxLength
	If Size <= 8000 varbinary
	else image (OLE Object)
xs:base64Binary	Same as xs:hexBinary
xs:integer	Number (Long Integer)
xs:int	Number (Long Integer)
xs:negativeInteger	Number (Long Integer); value constraint
xs:positiveInteger	Number (Long Integer); value constraint
xs:nonNegativeInteger	Number (Long Integer); value constraint
xs:nonPositiveInteger	Number (Long Integer); value constraint
xs:unsignedInt	Number (Long Integer)
xs:short	-- no equivalent --
xs:unsignedShort	-- no equivalent --
xs:long	-- no equivalent --
xs:unsignedLong	-- no equivalent --
xs:byte	Number (Byte)
xs:unsignedByte	Number (Byte)

26.4.2 MS SQL Server

MS SQL Server データベース(DB) がXML スキーマから生成される場合、MS SQL Server データ型は下のテーブルの通りXML スキーマデータ型に変換されます。

XML スキーマ データ型	MS SQL Server データ型
ID	uniqueidentifier
xs:string	If no facets
	{ if UNICODE nvarchar (255)
	else varchar (255) }
	else
	{ if UNICODE

	(Size = either length or maxLength)
	If Size <= 4000
	if FacetLengthIsSet then nChar
	else nVarChar
	if Size <= 1073741823 then nText }
	else
	{ if NON-UNICODE
	(Size = either length or maxLength)
	If Size <= 8000
	if FacetLengthIsSet then char
	else varchar
	if Size <= 2147483647 then text }
xs:normalizedString	Same as xs:string
xs:token	Same as xs:string
xs:Name	Same as xs:string
xs:NCName	Same as xs:string
xs:anyURi	Same as xs:string
xs:QName	Same as xs:string
xs:NOTATION	Same as xs:string
xs:boolean	bit
xs:float	real
xs:double	float
xs:decimal	decimal
xs:duration	datetime
xs:dateTime	datetime
xs:time	datetime
xs:date	datetime
xs:gYearMonth	datetime
xs:gYear	datetime
xs:gMonthDay	datetime
xs:gDay	datetime
xs:gMonth	datetime
xs:hexBinary	If no facets varbinary (255)
	(Size = either length or maxLength
	If Size <= 8000

(C) 2015-2021 Altova GmbH

	if FacetLengthIsSet then binary
	else varbinary
	if Size <= 2147483647 then image
xs:base64Binary	Same as xs:hexBinary
xs:integer	int
xs:int	int
xs:negativeInteger	Int (constrained to {...,-2,-1})
xs:positiveInteger	Int (constrained to {1,2,...})
xs:nonNegativeInteger	int (constrained to {0,1,2,...})
xs:nonPositiveInteger	int (constrained to {...,-2,-1,0})
xs:unsignedInt	int (additional constraints)
xs:short	smallint
xs:unsignedShort	smallint (additional constraints)
xs:long	bigint
xs:unsignedLong	bigint (additional constraints)
xs:byte	tinyint
xs:unsignedByte	tinyint (additional constraints)

26.4.3 MySQL

MySQL データベース(DB) がXML スキーマから生成される場合、MySQL データ型は下のテーブルの通りXML スキーマデータ型に変換されます。

XML スキーマ データ型	MySQL データ型
xs:ID	varchar(255)
xs:string	If no facets then varchar (255)
	else if facet length is set and <= 255 then char
	else if facet maxLength set and <= 255 then varchar
	else if maxLength is set and <= 65545 then text
	else if maxlength is set and <= 16777215 then mediumtext
	else if maxlength is set and <= 429496295 then longtext
xs:normalizedString	*Same as* xs:string

xs:token	*Same as* xs:string
xs:Name	*Same as* xs:string
xs:NCName	*Same as* xs:string
xs:anyURI	*Same as* xs:string
xs:QName	*Same as* xs:string
xs:NOTATION	*Same as* xs:string
xs:boolean	tinyint(1)
xs:float	float
xs:double	double
xs:decimal	decimal
xs:duration	timestamp
xs:dateTime	datetime
xs:time	time
xs:date	date
xs:gYearMonth	timestamp(4)
xs:gYear	year(4)
xs:gMonthDay	timestamp(8); constraints to check month, day
xs:gDay	timestamp(8); constraints to check day
xs:gMonth	timestamp(8); constraints to check month
xs:hexBinary	If no facets then blob (255)
	else if facet length is set and <= 255 then blob
	else if facet maxLength is set and <= 255 then tinyblob
	else if maxlength is set and <= 65545 then blob
	else if maxlength is set and <= 16777215 then mediumblob
	else if maxlength is set and <= 429496295 then longblob
xs:base64Binary	*Same as* xs:hexBinary
xs:integer	Integer
xs:int	int
xs:negativeInteger	Integer (constrained to {...,-2,-1})
xs:positiveInteger	Integer (constrained to {1,2,...})
xs:nonNegativeInteger	Integer (constrained to {0,1,2,...})

(C) 2015-2021 Altova GmbH

xs:nonPositiveInteger	Integer (constrained to {...,-2,-1,0})
xs:unsignedInt	Int (additional constraints)
xs:short	Smallint
xs:unsignedShort	Smallint (additional constraints)
xs:long	Bigint
xs:unsignedLong	Bigint (additional constraints)
xs:byte	Tinyint
xs:unsignedByte	Tinyint (additional constraints)

26.4.4 Oracle

Oracle データベース(DB) がXML スキーマから生成される場合、Oracle データ型は下のテーブルの通りXML スキーマデータ型に変換されます。

XML スキーマ データ型	Oracle データ型
xs:ID	ROWID
xs:string	If no facets
	if UNICODE then NVARCHAR2 (255)
	else VARCHAR2 (255)
	else if UNICODE
	(Size = either length or maxLength)
	If Size <= 2000 then NCHAR
	if Size <= 4000 then NVARHCAR2
	if Size <= 4 Gigabytes then NCLOB
	else if NON-UNICODE
	(Size = either length or maxLength)
	If Size <= 2000 then CHAR
	if Size <= 4000 then VARCHAR2
	if Size <= 4 Gigabytes then CLOB
xs:normalizedString	*Same as* xs:string
xs:token	*Same as* xs:string
xs:Name	*Same as* xs:string
xs:NCName	*Same as* xs:string
xs:anyURI	*Same as* xs:string
xs:QName	*Same as* xs:string

xs:NOTATION	*Same as* xs:string
xs:boolean	NUMBER with constraint Boolean
xs:float	FLOAT
xs:double	FLOAT
xs:decimal	NUMBER
xs:duration	TIMESTAMP
xs:dateTime	TIMESTAMP
xs:time	DATE
xs:date	DATE
xs:gYearMonth	INTERVAL YEAR TO MONTH
xs:gYear	DATE
xs:gMonthDay	DATE
xs:gDay	DATE
xs:gMonth	DATE
xs:hexBinary	if no facets then RAW (255)
	(Size = either length or maxLength)
	If Size <= 2000 then RAW (X)
	else Size <= 2 Gigabytes then LONG RAW (X)
	if Size <= 4 Gigabytes then BLOB (X)
xs:base64Binary	BLOB
xs:integer	NUMBER
xs:int	NUMBER
xs:negativeInteger	NUMBER (constrained to {...,-2,-1})
xs:positiveInteger	NUMBER (constrained to {1,2,...})
xs:nonNegativeInteger	NUMBER (constrained to {0,1,2,...})
xs:nonPositiveInteger	NUMBER (constrained to {...,-2,-1,0})
xs:unsignedInt	NUMBER (additional constraints)
xs:short	NUMBER
xs:unsignedShort	NUMBER (additional constraints)
xs:long	NUMBER
xs:unsignedLong	NUMBER (additional constraints)
xs:byte	BLOB
xs:unsignedByte	BLOB (additional constraints)

(C) 2015-2021 Altova GmbH

26.5 技術データ

このセクションは、ソフトウェアの技術面に関する役に立つ背景情報を含んでいます。以下のように整理されています:
OS とメモリ要件 1591 Altova XML バリデーター 1591 Altova XSLT とXQuery エンジン 1591 Unicode のサポート 1591 インターネットの使用 1592

26.5.1 OS とメモリ要件

オペレーティングシステム

Altova ソフトウェア アプリケーションは、以下のプラットフォームでご使用いただけます:
プラットフォーム更新済みのW indows 7 SP1、W indows 8、W indows 10プラットフォーム更新済みのW indows Server 2008 R2 SP1 または以降 **メモリ**
ソフトウェアがC++ で書かれているため、Java Runtime Environment をダウンロードする必要はなく、Java ベースのアプリケーションに比べ、通常少ないメモリを必要とします。しかしながら、各ドキュメントは完全に解析するため、また、ビューと編集の速度を向上するためにメモリにダウンロードされます。メモリの要件は、ドキュメントのサイズを増やします。

メモリ要件は、制限のない「元に戻す」履歴により影響を受けます。大きなドキュメントの大きなセクションの切り取り、貼り付け操作を繰り返し行うと、使用できるメモリがすぐに消費されます。

26.5.2 Altova XML バリデーター

XML ドキュメントを開くと、アプリケーションは、内蔵のXML バリデーターを使用して、指定されている場合、スキーマに対して整形式をチェックし、ツリーとインフォセットを作成します。XML バリデーターは、ドキュメントを編集する際にインテリジェントな編集ヘルプを提供し、発生する検証エラーを表示するために使用されます。

内蔵のXML バリデーターは、W 3C のXML スキーマ1.0 と1.1 仕様の最終勧告を実装しています。. New developments recommended by the W 3C XML スキーマ作業グループにより勧告される新しい項目は、XML バリデーターに継続的に組み込まれるため、Altova 製品は最高水準の開発環境を届けることができます。

26.5.3 Altova XSLT と XQuery エンジン

Altova 製品 は、Altova XSLT 1.0、2.0、および3.0 エンジンとAltova XQuery 1.0 と3.1 エンジンを使用しています。各エンジンのためのドキュメントと実装に固有の振る舞いに関しては、製品で使用されるエンジンの各ドキュメントの付属書（エンジン情報）で確認することができます。

メモ Altova MapForce は、XSLT 1.0、2.0 およびXQuery 1.0 エンジンを使用したコードを生成します。

26.5.4 Unicode のサポート

Altova XML 製品は、Unicode を完全にサポートします。XML ドキュメントを編集するには、ドキュメント内で使用されているUnicode 文字をサポートするフォントが必要です。

フォントの多くは、Unicode 範囲全体の特定のサブセットを含む場合があり、このため、通常は対応する表記システムをターゲットとします。テキストの一部が、文字化けして表示された場合、理由としては、選択されたフォントが必要とする字形を含まない場合があげられます。です

から、特に、異なる言語、または、異なる言語システムのXMLドキュメントを編集する場合、範囲全体をカバーするフォントを使用することが役にたちます。典型的なUnicode フォントは、Windows PC のArial Unicode MS で確認することができます。

アプリケーションフォルダーの/Examples フォルダー内で、異なる言語システムで表記された次の文章を含むUnicodeUTF-8.html という XHTML ファイルを確認してください:
When the world wants to talk, it speaks Unicode Wenn die Welt miteinander spricht, spricht sie Unicode 世界的に話すなら、Unicode です。) XHTML ファイルを開くと、Unicode の可能性を確認することができ、使用中の PC の使用することのできるフォントによりサポートされている表記システムが表示されます。

26.5.5 インターネットの使用

Altova アプリケーションは、次の状況でインターネット 接続を開始します:

- 登録ダイアログ(「**ヘルプ | ソフトウェアのライセンス認証**」)内の「評価キーコードをリクエスト」をクリックした場合、登録ダイアログボックス内の3つのフィールドが通常のhttp (ポート 80)接続を使用し、サーバーに転送され、無料の評価キーが顧客に通常の SMTP 電子メールを使用して送り返されます。
- Altova 製品の一部では、インターネットからファイルを開くことができます(「**ファイル | 開く | URL に切り替える**」)。この場合、ドキュメントは、次のプロトコルメソッドと接続の1つを使用して取得されます: HTTP (通常、ポート 80)、FTP (通常、ポート 20/21)、HTTPS (通常、ポート 443)。HTTP サーバーをポート 8080 で作動することもできます(URL ダイアログ内で、サーバー名とコロンの後にポートを指定します)。
- XML スキーマ、または、DTD を参照するXMLドキュメント、と、URL により指定されているドキュメントを開くと、参照されているスキーマドキュメントは、HTTP 接続(ポート 80) またはURL により指定されている他のプロトコル(上のポイント 2 参照)により抽出されます。XML ファイルが検証されている場合、スキーマドキュメントも抽出されます(オプションダイアログのファイルタブ内の(「**ツール | オプション**」))。アプリケーションに命令している場合、ドキュメントが開かれると検証が自動的に行われる場合もあります。
- WSDL とSOAP を使用するAltova アプリケーションでは、Web サービスを使用する接続は、WSDLドキュメントにより定義されています。
- XMLSpy 内で、「**電子メールで送信**」コマンドを使用する場合、(「**ファイル | 電子メールで送信**」) 現在選択されている範囲、または、ファイルは、ユーザーのマシンにインストールされているMAPI コンプライアント電子メールプログラムにより送信されます。
- ソフトウェアの荒いセンス認証 とLiveUpdate の一部として、Altova ソフトウェア使用許諾書内で更に詳しい説明を確認することができます。

(C) 2015-2021 Altova GmbH

26.6 ライセンス情報

このセクションには以下の内容が含まれています:

- ソフトウェアの配布 に関する情報
- ソフトウェアのアクティベーションとライセンスの計測
- ソフトウェアの使用に関する使用許諾契約書

本製品を使用する前に、上記の情報をよくお読みください。ソフトウェアのインストール時に上記のすべての条件に同意したとみなされ、お客様は上記の条件に拘束されることを同意したとみなされます。

Altova ライセンスの内容を確認するには、Altova Web サイト の Altova 法的な情報のページ に移動してください。

26.6.1 電子的なソフトウェアの配布

この製品は電子的なソフトウェアの配布により利用することが可能で、この配布方法により、以下のユニークなメリットがあります:

- 購入を決定する前に、無料でソフトウェアを試用することができます。*(Note: Altova MobileTogether Designer に対してライセンスを無料で割り当てることができます)。*
- Once ソフトウェアの購入を決定した際には、Altova Web サイト にて注文を行います。すぐにライセンス登録された製品の使用を開始することができます。
- オンラインにて注文を行うと、常に最新のソフトウェアをご利用いただけます。
- 製品パッケージには包括的なヘルプシステムが画面上に表示されます。最新バージョンのユーザーマニュアルは https://www.altova.com/ja/ 上にあり、(i) HTML フォーマットによる閲覧、ならびに (ii) PDF フォーマットのダウンロードと印刷に対応しております。

30 日間の評価期間

この製品をダウンロードした後は、最大で30日の間無料で製品の評価を行うことができます。20日間を超えた頃から、製品がライセンス登録されていないことがソフトウェアにより表示されます。このメッセージはアプリケーションが起動されるたびに表示され、30日間を超えてプログラムを使用するには、キーコードを含むライセンスファイルから提供される製品のライセンスを購入します。ライセンスファイルを製品のソフトウェアアクティベーションダイアログにアップロードして、製品をアンロックします。

https://shop.altova.com/ でライセンスを購入することができます

組織内でソフトウェアの評価を行う

評価版のソフトウェアを組織内のネットワークにて配布したい場合、またはインターネットに接続されていないコンピューターにてソフトウェアを使用する場合、どのような状態でも改変されていないことを条件に、セットアッププログラムだけの配布を行うことが可能です。ソフトウェアインストーラーへアクセスした人は、例外なく30日間の評価ライセンスキーコードをリクエストして、試用期間が経過した後は、製品を使い続けるためにライセンスの購入を行う必要があります。

26.6.2 ソフトウェアのアクティベーションとライセンスの計測

Altova のソフトウェアアクティベーションの一部として、ソフトウェアにより内部ネットワークまたはインターネットへの接続を行い、インストール時、登録時、Altova により使用されるライセンスサーバーの更新やライセンスの正当性を検証することで、ソフトウェアの不正な使用を防ぎ、顧客サービスを向上するため、ライセンスに関する情報を送信することがあります。アクティベーションにより、オペレーティングシステムや IP アドレス、

日付/時刻、ソフトウェアのバージョン、コンピュータの名前などのライセンスに関する情報が、お使いのコンピューターとAltova ライセンスサーバー間にてやり取りされます。

お使いのAltova 製品にはライセンス計測モジュールが内蔵されており、エンドユーザー使用許諾契約書の意図しない違反を防ぎます。お使いの製品はシングルユーザーまたはマルチユーザーとしてインストールされており、ライセンス計測モジュールにより、ライセンスされている数を超えたユーザーが同時に製品を使用することが無いことが保証されます。

このライセンス計測技術により、ローカルエリア接続（LAN）において、別々のコンピューター間で動作しているアプリケーションインスタンス間の通信が行われます。

シングルライセンス

ライセンス計測プロセスの一部としてアプリケーションが起動すると、ソフトウェアにより短いデータグラムがブロードキャストにより送信され、同一のネットワークセグメントにある他のコンピューターにてプログラムが動作していないかのチェックが行われます。応答が無い場合は、アプリケーションの他インスタンスから送信される信号に応えるため、ポートが開かれます。

マルチユーザーライセンス

同一のLAN 内にて2つ以上のアプリケーションインスタンスが使用された場合、スタートアップ時に、これらインスタンス間において通信が行われます。これらのインスタンス間にてキーコードのやりとりが行われ、購入された数のライセンスを超えてインスタンスが起動しないように保証することができます。このようなライセンス計測システムはUNIX やデータベース開発ツールにて広く使用されているもので、Altova ユーザーはリーズナブルな価格にて同時使用マルチユーザーライセンスを購入することができます。

弊社はアプリケーションのデザインも行っており、少数の小さなネットワークパケットを送信することで、ネットワークに対する負荷を最小限に抑えております。Altova により使用される2799番 TCP/IP ポートはIANA により公式登録されており（詳細は（ IANA W eb サイト (http://www.iana.org/) を参照ください）、弊社のライセンス計測モジュールも既にテストされたものです。

ファイヤーウォールを使用している場合、2799番ポートにて Altova 製品が動作しているコンピューター同士が通信しているのに気づかれるかも知れません。その他の手段によりライセンス使用許諾書の内容が守られることを保証できる限り、組織間の異なるグループにおいてこのようなトラフィックをブロックすることは勿論可能です。

証明書に関するメモ

Altova アプリケーションはHTTPS を介して Altova ライセンスサーバー（`link.altova.com`）に通信します。この通信のためにAltova は登録済みのSSL 証明書を使用します。（例えば、社内IT 部署または外部エージェンシーによりこの証明書が置き換えられている場合、使用中のAltova アプリケーションは接続が安全でないことを警告します。Altova アプリケーションを開始するために代替の証明書を使用することができますが、自己責任で行ってください。*安全ではない接続* の警告メッセージが表示されると、証明書の発行元を確認して（ Altova 証明書の代替証明書の使用の継続または停止を決定することができる）社内IT チームと相談してください。

（例えば、クライアントマシンへの、または、クライアントマシンへの通信を監視するために）自身の証明書の使用が必要な場合 Altova の無料管理ソフトウェアである Altova LicenseServer を使用中のネットワークにインストールすることが奨励されます。このセットアップでは、Altova LicenseServer はAltova との通信のためにAltova 証明書の使用を許可しつつクライアントマシンが所属機関の証明書の使用を継続することができます。

26.6.3 エンドユーザー使用許諾契約書

- Altova エンドユーザー使用許諾契約書: http://www.altova.com/ja/legal/eula
- Altova プライバシーポリシー: http://www.altova.com/ja/privacy

(C) 2015-2021 Altova GmbH

インデックス

, 230

.

.docx, 225, 560
.NET,
XMLSpy との統合, 729
スタンドアロン XMLSpy との違い, 731
と XMLSpy デバッガー, 733
.NET 拡張関数,
XSLT と XQuery, 1569
インスタンスメソッド、インスタンスフィールド, 1573
コンストラクター, 1571
データ型変換、.NET から XPath/XQuery へ, 1574
データ型変換、XPath/XQuery から .NET へ, 1574
概要, 1569
静的メソッド、静的フィールド, 1572
.NET 内のXSLT と XQuery のための拡張機能,
.NET 拡張関数を参照する, 1569
.pptx, 225, 560
.xslx, 225, 560

1

1行の長さ,
テキストビューにてワードラップ, 968

A

AAIDC ペイン, 187
ActiveX,
アプリケーションレベルの統合, 1416
ドキュメントレベルの統合, 1418
統合の必要条件, 1413
ActiveX controls,
support, 1082
ActiveX コントロール,
Visual Studio ツールボックスへの追加, 1414
ADO,
データ接続インターフェイスとして, 573
接続のセットアップ, 578
ADO.NET,
接続のセットアップ, 585
Altova XML パーサー,
について, 1591
Altova エンジン,
Altova 製品内で, 1591
Altova グローバルリソース,
グローバルリソース, 660
グローバルリソース を参照する, 660
Altova サポート, 117
Altova ソフトウェアの注文, 1049
Altova ソフトウェアの登録, 1049
Altova 拡張関数,
チャート関数 (チャート関数を参照), 1489
Altova 製品, 117
API,
documentation, 1095
JAVA, 1372
JAVA Classpath, 1372
overview, 1096
Application,
ActiveDocument, 1134
AddMacroMenuItem, 1134
AddXSLT_XQParameter, 1135
Application, 1135
ClearMacroMenu, 1135
CurrentProject, 1136
Dialogs, 1136
Documents, 1136
GetDatabaseImportElementList, 1137
GetDatabaseSettings, 1138
GetDatabaseTables, 1138
GetExportSettings, 1139
GetTextImportElementList, 1139
GetTextImportExportSettings, 1140
GetXSLT_XQParameterCount, 1140
GetXSLT_XQParameterName, 1141
GetXSLT_XQParameterXPath, 1141
ImportFromDatabase, 1141
ImportFromSchema, 1142
ImportFromText, 1143
ImportFromWord, 1144

Application,
NewProject, 1145
OnBeforeOpenDocument, 1132
OnBeforeOpenProject, 1133
OnDocumentOpened, 1133
OnProjectOpened, 1134
OpenProject, 1145
Parent, 1145
Quit, 1146
ReloadSettings, 1146
RemoveXSLT_XQParameter, 1146
RunMacro, 1147
ScriptingEnvironment, 1147
ShowApplication, 1148
ShowForm, 1148
URLDelete, 1149
URLMakeDirectory, 1149
WarningNumber, 1150
WarningText, 1150
Archive View, 225, 560
と EPUB ファイル, 568
と OOXML ファイル, 562
と ZIP ファイル, 566
ATL,
plug-in sample files, 1085
ATTLIST 宣言,
グリッドビューにて子として追加, 833
グリッドビューにて挿入, 818
グリッドビューにて追加, 826
グリッドビューにて変換, 836
AttributeFormDefault,
スキーマデザインビューにおける設定, 865
Authentic View, 433, 904
CDATA セクション, 410
DB データの編集, 902
GUI の概要, 415
PXF ファイルから出力ドキュメントを生成, 910
SPS テーブル, 432
XML テーブル, 433
XML テーブルアイコン, 437
XML テーブルの使用, 433
XML ドキュメントの印刷, 413
XML ドキュメントを開く, 404
XML/テキストとして貼り付け, 425
エンティティ, 410
エンティティの挿入, 412
コンテキストメニュー, 405
ツールバーアイコン, 416
データエントリデバイス, 410
データの挿入, 410
テーブル (SPS と XML), 432
テーブル内, 407
テキストの書式設定, 416
テキストの範囲, 425
ドキュメントの表示, 419
ノードの削除, 407
ノードの挿入, 407
ノードの追加, 407
マークアップタグの表示, 405
マークアップの表示, 416, 419
メインウィンドウ, 419
主要な機能の使用方法, 428
新たな XML ファイルを開く, 901
切り替え, 966
属性値の挿入, 411
特殊文字, 410
入力ヘルパー, 405, 421
要素のクリア, 407
要素の適用, 407
Authentic View テンプレート, 404
Authentic XML, 401
Authentic スクリプト,
セキュリティー設定, 910
信頼された場所, 910
Authentic スクリプトの信頼された場所, 910
Authentic ビュー,
XML 列の編集, 268
Authentic メニュー, 901
マークアップの表示, 416
動的なテーブルの編集, 416
AuthenticDataTransfer,
dropEffect, 1153
getData, 1153
ownDrag, 1153
type, 1153
AuthenticRange,
AppendRow, 1158
Application, 1159
CanPerformAction, 1159
CanPerformActionWith, 1160
Close, 1160
CollapsToBegin, 1160
CollapsToEnd, 1161
Copy, 1161
Cut, 1161
Delete, 1161

AuthenticRange,
DeleteRow, 1162
DuplicateRow, 1162
ExpandTo, 1163
FirstTextPosition, 1164
FirstXMLData, 1164
FirstXMLDataOffset, 1165
GetElementAttributeNames, 1166
GetElementAttributeValue, 1167
GetElementHierarchy, 1167
GetEntityNames, 1168
Goto, 1169
GotoNext, 1169
GotoNextCursorPosition, 1170
GotoPrevious, 1170
GotoPreviousCursorPosition, 1171
HasElementAttribute, 1171
InsertEntity, 1171
InsertRow, 1172
IsCopyEnabled, 1173
IsCutEnabled, 1173
IsDeleteEnabled, 1173
IsEmpty, 1173
IsEqual, 1174
IsFirstRow, 1174
IsInDynamicTable, 1174
IsLastRow, 1174
IsPasteEnabled, 1175
IsTextStateApplied, 1175
LastTextPosition, 1175
LastXMLData, 1176
LastXMLDataOffset, 1177
MoveBegin, 1178
MoveEnd, 1178
MoveRowDown, 1179
MoveRowUp, 1179
Parent, 1179
Paste, 1179
PerformAction, 1180
Select, 1181
SelectNext, 1181
SelectPrevious, 1182
SetElementAttributeValue, 1183
SetFromRange, 1184
Text, 1184
AuthenticView, 1202
Application, 1194
AsXMLString, 1195
DocumentBegin, 1196
DocumentEnd, 1196
Event, 1197
Goto, 1198
IsRedoEnabled, 1199
IsUndoEnabled, 1199
MarkupVisibility, 1200
OnBeforeCopy, 1186
OnBeforeCut, 1186
OnBeforeDelete, 1187
OnBeforeDrop, 1187
OnBeforePaste, 1188
OnDragOver, 1189
OnKeyboardEvent, 1190
OnMouseEvent, 1191
OnSelectionChanged, 1192
Parent, 1200
Print, 1200
Redo, 1201
Selection, 1201
Undo, 1202
WholeDocument, 1203
XMLDataRoot, 1203
Avro スキーマ, 462

C

C#,
XMLSpy の統合, 1421
CDATA,
グリッドビューにて子として追加, 829
グリッドビューにて挿入, 814
グリッドビューにて追加, 822
グリッドビューにて変換, 835
CDATA セクション,
Authentic View 内に挿入, 428
Class,
JAVA, 1372
Class ID,
XMLSpy 統合, 1425
ClassPath statement, 1372
CodeGeneratorDlg,
Application, 1204
CPPSettings_DOMType, 1205
CPPSettings_LibraryType, 1206
CPPSettings_UseMFC, 1206

CodeGeneratorDlg,
CSharpSettings_ProjectType, 1206
OutputPath, 1207
OutputPathDialogAction, 1207
OutputResultDialogAction, 1207
Parent, 1208
ProgrammingLanguage, 1208
PropertySheetDialogAction, 1208
TemplateFileName, 1209
COM API,
in Scripting Editor, 1065
COM-API,
documentation, 1095
Configure,
XMLSPY UI, 1082
CoreCatalog.xml, 302
CR&LF, 1015
CSS, 454
ドキュメントアウトライン, 457
プロパティ, 457
情報ウィンドウ, 457
文法色分け, 457
CSS 情報ウィンドウ, 457
CSV ファイル,
XML としてインポート, 938
CustomCatalog.xml, 302

D

Database connection,
setup examples, 603
DatabaseConnection,
ADOConnection, 1210
AsAttributes, 1210
CreateMissingTables, 1211
CreateNew, 1211
DatabaseKind, 1211
ExcludeKeys, 1212
File, 1212
IncludeEmptyElements, 1213
NumberDateTimeFormat, 1213
ODBCConnection, 1213
SQLSelect, 1214
TextFieldLen, 1215
DB, 440, 441
Authentic View 内のクエリ, 440
Authentic View 内の編集, 440, 445
Authentic View内のテーブルのナビゲート, 440
DB クエリ内のパラメーター, 441
クエリの作成, 441
内の Authentic Viewの表示のフィルター, 441
DB XML,
IBM DB2 にて XML スキーマを割り当て, 929
IBM DB2 に対して、XML スキーマの割り当て, 925
db2-fn:sqlquery, 351
db2-fn:xmlcolumn, 351
defaultOpenContent, 161
Delete,
Application.URLDelete, 1149
DF,
XMLSpy における変換, 323
Dialogs,
Application, 1215
CodeGeneratorDlg, 1216
DTDSchemaGeneratorDlg, 1217
FileSelectionDlg, 1216
GenerateSampleXMLDlg, 1217
Parent, 1216
SchemaDocumentationDlg, 1217
directories,
creating with Application.URLMakeDirectory, 1149
DocEditEvent (obsolete),
altKey (obsolete), 1322
altLeft (obsolete), 1323
button (obsolete), 1323
cancelBubble (obsolete), 1324
clientX (obsolete), 1325
clientY (obsolete), 1326
ctrlKey (obsolete), 1326
ctrlLeft (obsolete), 1327
dataTransfer (obsolete), 1328
fromElement (obsolete), 1329
keyCode (obsolete), 1329
propertyName (obsolete), 1330
repeat (obsolete), 1330
returnValue (obsolete), 1330
shiftKey (obsolete), 1331
shiftLeft (obsolete), 1331
srcElement (obsolete), 1332
type (obsolete), 1333
DocEditView (obsolete),
ApplyTextState (obsolete), 1337
CurrentSelection (obsolete), 1338
EditClear (obsolete), 1338

DocEditView (obsolete),
EditCopy (obsolete), 1339
EditCut (obsolete), 1339
EditPaste (obsolete), 1340
EditRedo (obsolete), 1340
EditSelectAll (obsolete), 1340
EditUndo (obsolete), 1341
event (obsolete), 1341
GetAllowedElements (obsolete), 1342
GetNextVisible (obsolete), 1344
GetPreviousVisible (obsolete), 1344
IsEditClearEnabled (obsolete), 1345
IsEditCopyEnabled (obsolete), 1345
IsEditCutEnabled (obsolete), 1346
IsEditPasteEnabled (obsolete), 1346
IsEditRedoEnabled (obsolete), 1347
IsEditUndoEnabled (obsolete), 1347
IsRowAppendEnabled (obsolete), 1348
IsRowDeleteEnabled (obsolete), 1348
IsRowDuplicateEnabled (obsolete), 1349
IsRowInsertEnabled (obsolete), 1349
IsRowMoveDOwnEnabled (obsolete), 1350
IsRowMoveUpEnabled (obsolete), 1350
IsTextStateApplied (obsolete), 1351
IsTextStateEnabled (obsolete), 1351
LoadXML (obsolete), 1352
RowAppend (obsolete), 1353
RowDelete (obsolete), 1353
RowDuplicate (obsolete), 1354
RowInsert (obsolete), 1354
RowMoveDown (obsolete), 1355
RowMoveUp (obsolete), 1355
SaveXML (obsolete), 1356
SelectionMoveTabOrder (obsolete), 1356
SelectionSet (obsolete), 1357
XMLRoot (obsolete), 1358
DOCTYPE 宣言,
グリッドビューにて子として追加, 832
グリッドビューにて追加, 817, 825
グリッドビューにて変換, 836
Document, 1229
Application, 1222
AssignDTD, 1222
AssignSchema, 1223
AssignXSL, 1223
AssignXSLFO, 1223
AuthenticView, 1224
Close, 1225
ConvertDTDOrSchema, 1225
CreateChild, 1227
CreateSchemaDiagram, 1227
CurrentViewMode, 1228
DataRoot, 1228
DocEditView, 1228
Encoding, 1229
EndChanges, 1229
ExecuteXQuery, 1229
ExportToDatabase, 1230
ExportToText, 1231
FullName, 1232
GenerateDTDOrSchema, 1233
GenerateProgramCode, 1234
GenerateSampleXML, 1234
GenerateSchemaDocumentation, 1234
GetExportElementList, 1235
GetPathName, 1236
GridView, 1236
IsModified, 1236
IsValid, 1236
IsWellFormed, 1239
Name, 1239
OnBeforeCloseDocument, 1220
OnBeforeSaveDocument, 1220
OnBeforeValidate, 1221
OnCloseDocument, 1221
OnViewActivation, 1222
Path, 1240
RootElement, 1240
Save, 1240
SaveAs, 1241
Saved, 1241
SaveInString, 1241
SaveToURL, 1242
SetActiveDocument, 1242
SetEncoding, 1242
SetExternalIsValid, 1243
SetPathName, 1243
StartChanges, 1244
SwitchViewMode, 1244
Title, 1245
TransformXSL, 1245
TransformXSLFO, 1245
UpdateViews, 1246
UpdateXMLData, 1246
XQuery, 1229
Documents,

Documents,
Count, 1247
Item, 1248
NewAuthenticFile, 1248
NewFile, 1248
NewFileFromText, 1249
OpenAuthenticFile, 1249
OpenFile, 1249
OpenURL, 1250
OpenURLDialog, 1250
DTD, 281, 1015, 1018
ATTLIST 宣言, 818, 826, 833, 836
ELEMENT 宣言, 818, 826, 832, 836
ENTITY 宣言, 818, 826, 833, 836
NOTATION 宣言, 819, 827, 833, 836
UML へ変換, 859
XML から外部 DTD への参照, 817
XML スキーマから生成(Enterprise ならびに Professional エディションのみ), 284
XML スキーマに変換する, 856
XML スキーマへの変換(Enterprise と Professional エディション), 282
XML ドキュメントから移動, 853
XML ドキュメントから生成, 854
XML ドキュメントから定義へ移動, 854
XML ドキュメントへ割り当て, 852, 853
XML ファイルから生成する, 282
XML ファイルのアウトラインを生成する, 860
エンティティの読み込み, 853
グリッドビュー内の編集 (Enterprise と Professional エディション), 282
テキストビュー内での編集, 282
関連するメニューコマンド, 852
DTD とカタログ, 301
DTD における ELEMENT 宣言,
グリッドビューにて挿入, 818
DTD/スキーマメニュー, 852
DTDSchemaGeneratorDlg,
Application, 1251
AttributeTypeDefinition, 1252
DTDSchemaFormat, 1252
FrequentElements, 1252
GlobalAttributes, 1252
MaxEnumLength, 1253
MergeAllEqualNamed, 1253
OnlyStringEnums, 1253
OutputPath, 1253
OutputPathDialogAction, 1254
Parent, 1254
ResolveEntities, 1254
TypeDetection, 1255
ValueList, 1255

E

Eclipse における XMLSpy パースペクティブ, 740
Eclipse のための XMLSpy プラグイン,
インストール, 735
Eclipse プラットフォーム,
Eclipse における, 737
XMLSpy デバッガーパースペクティブ, 740
と XMLSpy, 734
と XMLSpy Integration Package, 735
Eclipse 内部で XMLSpy を使用する, 734
ELEMENT 宣言,
グリッドビューにて子として追加, 832
グリッドビューにて追加, 826
グリッドビューにて変換, 836
ElementFormDefault,
スキーマデザインビューにおける設定, 865
ElementList,
Count, 1256
Item, 1256
RemoveElement, 1256
ElementListItem,
ElementKind, 1256
FieldCount, 1257
Name, 1257
RecordCount, 1257
Enter キー, 451
Enter キー参照, 451
使用の効果, 451
ENTITY 宣言,
グリッドビューにて子として追加, 833
グリッドビューにて挿入, 818
グリッドビューにて追加, 826
グリッドビューにて変換, 836
Enumerations,
SPYAttributeTypeDefinition, 1358
SPYAuthenticActions, 1359
SPYAuthenticDocumentPosition, 1359
SpyAuthenticElementActions, 1359
SPYAuthenticElementKind, 1360
SPYAuthenticMarkupVisibility, 1360
SPYDatabaseKind, 1360

Enumerations,
SPYDialogAction, 1361
SPYDOMType, 1361
SPYDTDSchemaFormat, 1361
SPYEncodingByteOrder, 1362
SPYExportNamespace, 1362
SPYFrequentElements, 1362
SPYKeyEvent, 1363
SPYLibType, 1364
SPYLoading, 1364
SPYMouseEvent, 1364
SPYNumberDateTimeFormat, 1365
SPYProgrammingLanguage, 1365
SPYProjectItemTypes, 1365
SPYProjectType, 1366
SPYSampleXMLGenerationOptimization, 1366
SPYSampleXMLGenerationSchemaOrDTDAssignment, 1367
SPYSchemaDefKind, 1367
SPYSchemaDocumentationFormat, 1368
SPYTextDelimiters, 1369
SPYTextEnclosing, 1369
SPYTypeDetection, 1369
SPYURLTapes, 1369
SPYViewModes, 1370
SPYVirtualKeyMask, 1371
SPYXMLDataKind, 1371
EPUB ファイル, 568
Event, 1132, 1133, 1134, 1186, 1187, 1188, 1189, 1190, 1191, 1192, 1220, 1221, 1222, 1279, 1280, 1281
Events, 1102
Excel 2007, 225, 560
ExportSettings,
CreateKeys, 1258
ElementList, 1258
EntitiesToText, 1258
ExportAllElements, 1258
FromAttributes, 1259
FromSingleSubElements, 1259
FromTextValues, 1259
IndependentPrimaryKey, 1259
Namespace, 1259
SubLevelLimit, 1260
External ID 宣言,
グリッドビューにて子として追加, 832
グリッドビューにて挿入, 817
グリッドビューにて追加, 825
グリッドビューにて変換, 836

F

FileSelectionDlg,
Application, 1260
DialogAction, 1260
FullName, 1261
Parent, 1261
Firebird,
ODBC を使用しての接続, 605
FO に対するXSLT スタイルシート,
XML ドキュメントへ割り当て, 894
FOP,
フォント, 277

G

Generate Sample XML, 1358, 1366, 1367
GenerateSampleXMLDlg,
Application, 1274
FillWithSampleData, 1275
NonMandatoryAttributes, 1276
NonMandatoryElements, 1276
Parent, 1277
RepeatCount, 1277
TakeFirstChoice, 1278
GridView,
CurrentFocus, 1281
Deselect, 1282
IsVisible, 1282
OnBeforeDrag, 1279
OnBeforeDrop, 1280
OnBeforeStartEditing, 1280
OnEditingFinished, 1281
OnFocusChanged, 1281
Select, 1282
SetFocus, 1282
GUI 詳細, 103

H

HTML, 454
XMLSpy の統合, 1427

HTML サンプル,
of XMLSpyControl 統合, 1424
XMLSpyControl 統合, 1425, 1426
HTML ドキュメント,
情報ウィンドウ, 455
編集, 455
HTML 出力,
PXF ファイルからAuthentic View 内に生成, 910
HTML 情報 ウィンドウ, 455
HTTP, 550
HTTP メッセージウィンドウ,
HTTP メソッド, 551
HTTP 状態コード, 557
と WADL, 555
ヘッダーの受け入れ, 557
リクエストのインポート, 555
リクエストの送信, 551
レスポンスの受け入れ, 557

I

IBM DB2,
JDBC を介した接続, 607
ODBC を使用しての接続, 609
XML スキーマの割り当て, 925
XML スキーマを XML ファイルへ割り当て, 929
スキーマ管理ならびに割り当て, 925
IBM DB2 for i,
ODBC を使用しての接続, 617
IBM Informix,
connecting through JDBC, 616
JDBC を使用しての接続, 620
Insert コマンド,
グリッドビューにて, 812

J

Java, 1431
API, 1372
ClassPath, 1372
Java 仮想マシン,
パス設定, 1039
Java 拡張関数,
XSLT と XQuery, 1561
インスタンスメソッド、インスタンスフィールド, 1567
コンストラクター, 1566
データ型変換、 Java から XPath/XQuery へ, 1569
データ型変換、XPath/XQuery から Java へ, 1568
ユーザー定義の JAR ファイル, 1565
ユーザー定義のクラスファイル, 1563
概要, 1561
静的メソッド、静的フィールド, 1567
Java 設定, 1039
JDBC,
Teradata への接続, 652
データ接続インターフェイスとして, 573
接続の設定 (Windows), 594
JRE,
Eclipse のための XMLSpy プラグイン, 735
JScript,
scripting with XMLSpy, 1056
JSON, 454
JSON インスタンスを XML インスタンスに変換, 959
XML スキーマ を JSON スキーマに変換する(JSON スキーマから XML スキーマに変換する), 962
XQuery 式, 543
と XPath, 541
と XPath/XQuery 出力ウィンドウ, 541
と XQuery, 541
JSON グリッドビュー, 475
XQuery 式, 492, 493
イメージの表示, 497
オブジェクト内でコンポーネントをラップ, 849
コンテキストメニューコマンド, 500
コンポーネントの挿入, 849
コンポーネントの追加, 849
セル内のコマンドボタン, 481, 482
チャートの作成, 498
テーブルアイコン, 488
テーブルの行と列をフリップする, 851
テーブルの表示, 488
テーブルの表示コマンド, 851
テーブル内の列を並べ替える, 851
ドキュメントの表示, 478
ドキュメントの編集, 482
ドキュメント構造の編集, 481
ドラッグアンドドロップ, 485
ビューのフィルター, 492
メタデータファイル, 492, 493
異なる型のシンボル, 478
型コマンド, 848
子としてコンポーネントを追加する, 849

JSON グリッドビュー, 475
自動補完, 482
出力を計算するためのフォーミュラ, 492, 493
配列内でコンポーネントをラップ, 849
JSON グリッドビュー内でのドラッグとドロップ, 485
JSON グリッドビュー内の XQuery 式, 492
JSON グリッドビュー内のイメージ, 497
JSON グリッドビュー内のチャート, 498
JSON グリッドビュー内のテーブル,
TSV または JSON としてコピー, 502
JSON グリッドビュー内のテーブル表示,
スプレッドシートのインポート/エクスポート, 488
編集, 488
JSON グリッドビュー内のフィルター, 492
の設定, 502
JSON グリッドビュー内のフォーミュラ, 492, 493
の設定, 502
JSON スキーマ, 462, 469
allOf, 530
any, 527
anyOf, 530
if-then-else, 531
JSON インスタンスからの生成, 545
multiple, 527
not, 530
oneOf, 530
アトミック方, 526
オブジェクト, 515, 517
オブジェクトと依存性, 521
オブジェクトプロパティ, 515
演算子, 530
基本型, 526
型セレクター(any, multiple), 527
指定されていないプロパティ, 517
条件, 531
数値定義, 526
説明, 467
単純型, 526
配列, 524
文字列定義, 526
用語, 467
JSON スキーマバージョン, 506
JSON スキーマビュー, 505
デザインビュー, 513
外部スキーマの追加, 509
構成, 532
詳細入力ヘルパー, 509
制約入力ヘルパー, 509
入力ヘルパー, 509
JSON データ,
オブジェクト, 464
サンプル, 464
型, 464
配列, 464
JSON ドキュメント,
XML から、または、へ変換, 549
XML からの変換, 470
XMLSpy 内で開く, 462
XQuery式のための, 397
グリッドビュー内での編集, 475
テキストビューでの編集, 470
検証, 537
新規作成, 462
JSON ドキュメントのグリッドビュー,
JSON グリッドビューを参照, 475
JSON ドキュメントの構文チェック, 470, 475
JSON ドキュメントの整形式チェック, 470
JSON ドキュメントの整形式のチェック, 475
JSON のための XQuery 式, 543
JSON メニュー, 848
JSON ライン, 469

L

loading, 1250

M

Macros,
developing, 1056, 1061
enabling, 1067, 1079
running, 1080
MapForce, 860
MariaDB,
ODBC を使用した接続, 622
Microsoft Access,
ADO を介しての接続, 578, 624
Microsoft Office 2007, 225, 560
Microsoft SQL Server,
ADO を介しての接続, 627
ODBC を使用しての接続, 629
Microsoft® SharePoint® Server, 803

MIME, 1018
MS SQL Server,
スキーマ拡張, 879
スキーマ設定, 880
MSXML, 1036
MSXSL スクリプト内の拡張関数, 1575
msxsl:script, 1575
MySQL,
ODBC を使用しての接続, 635

N

NOTATION 宣言,
グリッドビューにて子として追加, 833
グリッドビューにて挿入, 819
グリッドビューにて追加, 827
グリッドビューにて変換, 836

O

OASIS,
XML カタログ, 841
ODBC,
MariaDB への接続, 622
Teradata への接続, 654
データ接続インターフェイスとして, 573
接続のセットアップ, 591
ODBC ドライバー,
使用可能か確認する, 591
Office Open XML, 225, 560
OLE DB,
データ接続インターフェイスとして, 573
OOXML,
Office Open XML を参照, 225
Office Open XML 以下を参照, 560
Open Office XML,
Archive View にて作成, 562
Archive View にて編集, 562
サンプルファイル, 564
openContent, 161
Oracle,
スキーマ拡張, 878
スキーマ設定, 879
Oracle XML Db, 933
データベースの参照, 936
Oracle XML データベース,
XML スキーマの管理, 933
Oracle データベース,
JDBC を使用しての接続, 637
ODBC を使用しての接続, 638
OS,
Altova 製品のための, 1591
Overview,
of XMLSpy API, 1096

P

Parent, 1239
PDF フォント, 277
PDF 出力,
PXF ファイルからAuthentic View 内に生成, 910
Plug-in,
ATL sample files, 1085
registration, 1081
User interface configuration, 1082
XMLSPY, 1081
PostgreSQL,
ODBC を使用しての接続, 644
直接 (ネイティブ)接続, 596
直接接続する(ネイティブ), 598
PowerPoint 2007, 225, 560
Progress OpenEdge データベース,
JDBC を使用しての接続, 647
ODBC を使用しての接続, 648
PUBLIC,
識別子 - カタログ, 841
PXF ファイル,
Authentic View から出力ドキュメントを生成, 910

R

Register,
plug-in, 1081
Return キー,
Enter キーを参照, 451
RootCatalog.xml, 302
RTF 出力,
PXF ファイルからAuthentic View 内に生成, 910

S

save, 1242
schema, 1142
SchemaAgent, 298
XMLSpy からサーバーへ接続, 881
XMLSpy からスキーマを開く, 298
サーバーから切断, 882
スキーマの表示, 882
拡張検証, 882
作業する, 292, 293, 298
SchemaAgent Server,
接続, 290
SchemaAgent Server へ接続, 290, 881
SchemaDocumentationDlg,
AllDetails, 1283
Application, 1284
IncludeAll, 1285
IncludeAttributeGroups, 1286
IncludeComplexTypes, 1286
IncludeGlobalElements, 1287
IncludeGroups, 1287
IncludeIndex, 1287
IncludeLocalElements, 1288
IncludeRedefines, 1288
IncludeSimpleTypes, 1289
OptionsDialogAction, 1289
OutputFile, 1290
OutputFileDialogAction, 1290
OutputFormat, 1290
Parent, 1290
ShowAnnotations, 1291
ShowAttributes, 1291
ShowChildren, 1291
ShowConstraints, 1292
ShowDiagram, 1292
ShowEnumerations, 1292
ShowNamespace, 1293
ShowPatterns, 1293
ShowProgressBar, 1293
ShowProperties, 1293
ShowResult, 1294
ShowSingleFacets, 1294
ShowSourceCode, 1294
ShowType, 1295
ShowUsedBy, 1295
Scripting Editor,
overview, 1056, 1058
SharePoint® Server, 803
SOAP デバッガー,
Visual Studio .NET 内の SOAP デバッガー, 733
Source control,
installing a source-control plug-in, 704
SPP ファイルの位置, 782
SPS,
新たな XML ファイルに関連付ける, 742
SPS テーブル,
動的なテーブルの編集, 416
SPS テーブル 内の Authentic View,
使用方法, 432
SPS ファイル,
XML ファイルに割り当てる, 903
SpyProject,
CloseProject, 1296
ProjectFile, 1296
RootItems, 1297
SaveProject, 1297
SaveProjectAs, 1297
SpyProjectItem,
ChildItems, 1298
FileExtensions, 1298
ItemType, 1298
Name, 1298
Open, 1298
ParentItem, 1298
Path, 1299
ValidateWith, 1299
XMLForXSLTransformation, 1299
XSLForXMLTransformation, 1299
XSLTransformationFileExtension, 1299
XSLTransformationFolder, 1299
SpyProjectItems,
AddFile, 1300
AddFolder, 1300
AddURL, 1300
Count, 1301
Item, 1301
RemoveItem, 1301
SQL Server,
ADO.NET を使用して接続する, 585
XML スキーマの管理, 931
SQL エディター,
クエリの作成, 922

SQL エディター,
その説明, 920
データベースクエリウィンドウにて, 920
SQL サーバー,
ADO を介しての接続, 578
SQL スクリプトにおけるブックマーク, 920
SQL スクリプトにおける自動補完, 920
SQL スクリプト内のコマンド, 920
SQL スクリプト内の領域, 920
SQLite,
外部キーの無効化, 601
接続のセットアップ (Windows), 601
StyleVision, 860
StyleVision Power Stylesheetの編集, 903
StyleVision Power Stylesheet,
StyleVision内の編集, 903
XML ファイルへ割り当てる, 903
StyleVision Power Stylesheet を XML ファイルに割り当てる, 903
Sybase,
JDBC を使用しての接続, 651

T

Teradata,
JDBC を使用した接続, 652, 654
terminate, 1146
TextImportExportSettings,
DestinationFolder, 1302
EnclosingCharacter, 1302
Encoding, 1302
EncodingByteOrder, 1302
FieldDelimiter, 1303
FileExtension, 1303
HeaderRow, 1303
ImportFile, 1303
TSV,
JSON グリッドビューテーブルを以下としてコピー, 502
XML グリッドビューテーブルを以下としてコピー, 265

U

UCS-2, 1020
UML,
スキーマから変換, 859
Unicode のサポート,
Altova 製品内で, 1591
URL, 1149, 1242, 1250
メールで送信, 757
User interface,
configure using plug-in, 1082
UTF-16, 1020

V

VBScript,
scripting with XMLSpy, 1056
Visual Studio,
XMLSpy ActiveX コントロールをツールボックスへ追加, 1414
Visual Studio .Net,
と XMLSpy, 729
と XMLSpy デバッガー, 733
と XMLSpy の違い, 731
VS .NET,
と XMLSpy 統合パッケージ, 730
VS .NET に対応した XMLSpy プラグイン,
インストール, 730

W

WADL,
HTTP リクエストのために使用, 555
Web サーバー, 1052
Windows,
Altova 製品のためのサポート, 1591
Word 2007, 225, 560
Word 2007+ 出力,
PXF ファイルからAuthentic View 内に生成, 910
Word ドキュメント,
XML としてインポート, 946
WSDL ファイル,
拡張性要素, 846
検証, 846

X

XInclude,

XInclude,
XML ドキュメントの挿入, 234
グリッドビューにて子として追加, 830
グリッドビューにて挿入, 815
グリッドビューにて追加, 768, 823
テキストビューにて挿入, 768
XML,
Oasis カタログ, 841
スペルチェッカー, 974
構造化テキストとしてコピー, 764
XML DB,
新たにデータ行を Authentic View へロード, 904
新規 XML データ行のロード, 440
XML DB 内のデータを編集, 904
XML インポート,
スキーマをベースに, 950
XML グリッドビュー,
グリッドビューを参照, 236
XML グリッドビュー内でのドラッグとドロップ, 246
XML グリッドビュー内の XQuery 式, 254, 255
XML グリッドビュー内のテーブル,
TSV または XML としてコピー, 265
XML グリッドビュー内のテーブル表示,
スプレッドシートのインポート/エクスポート, 250
編集, 250
XML グリッドビュー内のフィルター, 254
の設定, 265
XML スキーマ, 41, 281
<alternative> 要素, 181
DB XML へ割り当て, 929
DTD, 232
DTD スキーマの生成(Enterprise と Professional エディション), 282
DTD に変換する, 856
IBM DB2 データベースにおける管理と割り当て, 925
IBM DB2 にて管理, 925
MS SQL Server スキーマ設定, 880
MS SQL Server 拡張, 879
Oracle スキーマ設定, 879
Oracle 拡張, 878
XML スキーマへの変換(Enterprise ならびに Professional エディションのみ), 284
XML ドキュメントから移動, 854
XML ドキュメントから生成, 854
XML ドキュメントから定義へ移動, 854
XML ドキュメントの生成, 284
XML ドキュメントを編集中に修正する, 93
XML ファイルのアウトラインを生成する, 860
インクルードされたスキーマのフラット化, 285
インクルードされたスキーマをフラット化, 884
グリッドビューで編集(Enterprise ならびに Professional エディションのみ), 284
グローバル コンポーネント, 161
グローバルリソース, 232
コンテンツ モデルの編集, 168
コンテンツモデルダイアグラム, 868
コンテンツモデルビューの設定, 873
コンポーネンツの追加, 44
サブセットへ分割, 883
スキーマデザインビューにおける設定, 865
スキーマデザインビューにおける名前空間の設定, 865
スキーマも参照, 852
スマートフィックス, 211
チュートリアル, 41
テキストビューで編集, 284
デザインビューでのナビゲーション, 68
ドキュメンテーションの生成, 868
ビューの構成, 50
関連するメニューコマンド, 852
基本的なスキーマの作成, 41
検証, 211
新しいファイルの作成, 41
他のスキーマを含める, 285
名前空間の定義, 43
要素を追加する, 49
XML スキーマのスマートフィックス, 211
XML スキーマの検証, 211
XML スキーマ定義,
高度な機能, 56
XML データ,
データベースへエクスポート, 957
テキストファイルへエクスポート, 954
XML データをデータベースへエクスポート, 957
XML データをテキストファイルへエクスポート, 954
XML テーブル 内の Authentic View,
編集のためのアイコン, 437
XML テーブル内の Authentic View,
使用方法, 433
XML テキスト, 1021
コピー, 764
XML テキストのコメントアウトとアンコメント, 781
XML ドキュメント, 229
Authentic View 内で開く, 404
Authentic ビュー内での編集, 268
DTD から生成, 282
Oracle XML Db の参照, 936
XInclude の挿入, 234

XML ドキュメント, 229
XML スキーマから生成(Enterprise ならびに Professional エディションのみ), 284
XPath 条件式の評価, 279
XQuery 実行の自動化, 275
XSLT スタイルシートへ割り当て, 894
XSLT による変換, 275
XSLT 変換の自動化, 275
エンコーディング, 279
グリッドビュー内での編集, 236
スキーマの生成, 279
スキーマを割り当て(DTD を含む), 232
テキストのインポートとエクスポート, 279
テキストビュー内での編集, 79
デフォルトビュー, 230
と XQuery, 275
とテキストビュー内のコメント, 234
とテキストビュー内の空のライン, 234
とノードの XPath 式, 234
のためのテキストビュー編集機能, 234
の妥当性のチェック, 83
ファイルパスの挿入, 234
開く, 230
検索と置き換え, 279
検証, 232
自動検証, 230
新規作成, 74
整形式のチェック, 232
保存, 230
XML ドキュメントで選択されたノードの XPath,
クリップボードにコピー, 768
XML ドキュメントで選択されたノードの XPointer,
クリップボードにコピー, 768
XML ドキュメントの検証, 232
XML ドキュメントの整形式をチェック, 232
XML ドキュメント作成,
チュートリアル, 74
XML の差分,
ディレクトリの比較, 987
ファイルの比較, 990
XML パーサー,
について, 1591
XML ファイル,
DTD または XML スキーマから生成する, 860
XML プロローグ,
グリッドビューにて子として追加, 829
グリッドビューにて挿入, 814
グリッドビューにて追加, 822
グリッドビューにて変換, 835
XML メニュー, 812
XML 差分, 983
ファイルの比較, 983
XML 署名, 449, 905
xml:base, 220
xml:id, 220
xml:lang, 220
xml:space, 220
XMLData,
AppendChild, 1311
EraseAllChildren, 1312
EraseCurrentChild, 1312
GetChild, 1313
GetChildKind, 1314
GetCurrentChild, 1315
GetFirstChild, 1315
GetNextChild, 1316
HasChildren, 1317
HasChildrenKind, 1317
InsertChild, 1318
IsSameNode, 1319
Kind, 1319
MayHaveChildren, 1319
Name, 1320
Parent, 1320
TextValue, 1320
XMLSpy, 741
plug-in registration, 1081
ヘルプ, 117
機能, 117
統合, 1413
XMLSpy API,
documentation, 1095
overview, 1096
XMLSpy Enterprise Edition,
ユーザーマニュアル, 22
XMLSpy Integration Package, 735
XMLSPY plug-in, 1081
XMLSpy コマンドテーブル, 1440
XMLSpy でのプロジェクト,
の利点, 98
作成方法, 98
XMLSpy でのプロジェクト管理, 98
XMLSpy における SchemaAgent, 289
XMLSpy におけるデータベース, 572
XMLSpy を SchemaAgentサーバーから切断, 882
XMLSPY を使用して編集, 1015

XMLSpy 統合,
のサンプル, 1424, 1425, 1426
XMLSpy 統合パッケージ, 730
XMLSpy 内の XQuery Update, 345
XMLSpy 内のカタログ, 302
XMLSpyControl, 1462
C# を使用した統合, 1421
HTML を使用下統合, 1427
アプリケーションレベルでのサンプル 統合, 1424, 1425, 1426
オブジェクトレファレンス, 1459
ドキュメント, 1413
ドキュメントレベルでの統合のサンプルサンプル, 1421
XMLSpyControl ドキュメント, 1470
XMLSpyControl プレースホルダ, 1476
XMLSpyDocumentEditor,
MarkUpView, 1352
XMLSpyLib, 1095, 1096
Application, 1131
AuthenticDataTransfer, 1152
AuthenticRange, 1157
AuthenticSelection (obsolete), 1333
AuthenticView, 1185
CodeGeneratorDlg, 1203
DatabaseConnection, 1209
Dialogs, 1215
DocEditEvent (obsolete), 1321
DocEditView (obsolete), 1335
Document, 1218
Documents, 1246
DTDSchemaGeneratorDlg, 1251
ElementList, 1255
ElementListItem, 1256
ExportSettings, 1257
FileSelectionDlg, 1260
GenerateSampleXMLDlg, 1273
GridView, 1279
ProjectItem, 1297
SchemaDocumentationDlg, 1282
SpyProject, 1296
SpyProjectItems, 1300
TextImportExportSettings, 1301
XMLData, 1309
XMLSpyコマンド,
XMLSpyControl, 1459
XMLSpyControl 内, 1461
XMLグリッドビュー内のフォーミュラ, 255
の設定, 265
XML-準拠, 1018
XPath,
XML ドキュメント内でノードを生成する方法, 234
評価, 108
XPath 1.0,
XPath の評価, 839
XPath 2.0,
XPath の評価, 839
XPath ウィンドウ, 108
XPath のコピー, 768
XPath の評価, 108
使用方法, 839
XPath/XQuery ウィンドウ, 381, 382
XPath/XQuery デバッガー内のウォッチ式, 389
XPath/XQuery デバッガー内のデバッグポイント, 389
XPath/XQuery デバッガー内のトレースポイント, 389
XPath/XQuery デバッガー内のブレークポイント, 389
XPath/XQuery デバッガー内の変数, 389
XPath/XQuery 式,
JSON ドキュメントのための, 397
XPath/XQuery ウィンドウ, 399
XPath/XQuery ウィンドウ内での作成, 395
デバッグ, 389
評価, 384
評価とデバッグ, 381
XPath/XQuery 式のデバッグ, 389
XPath/XQueryデバッガー内のブレークポイント, 389
XPaths,
トレースポイントの設定, 375
XPath-Watch ウィンドウ,
XSLT/XQuery デバッガー, 367
XPointer,
XML ドキュメント内でノードを生成する方法, 234
XPointer のコピー, 768
XQuery, 336, 342
DB サポート, 351
IBM DB2 に対する関数, 351
XML データベースの問い合わせ, 351
Xquery ドキュメントに変数をパスする, 890
インテリジェントな編集機能, 338
テキストビューにて編集, 332
ドキュメント検証, 342
ファイルを開く, 334
拡張関数, 1561
実行, 342
入力ヘルパー, 335
XQuery Update,
PUL, 345

XQuery Update,
プレビュー, 345
XQuery Update Facility, 345
XQuery Update オプション, 1038
XQuery Update 内のPUL, 345
XQuery オプション, 1038
XQuery デバッガー,
XSLT/XQuery デバッガーを参照, 356
XQuery ファイル,
XMLSpy にてファイル拡張子をセット, 334
XQuery プロセッサー,
Altova 製品内で, 1591
XQuery 更新, 343
XQuery 実行, 343, 893
XSD 1.0 と 1.1,
編集モード, 153
XSD モード, 153
XSD 検証, 211
xsi:type,
使用方法, 76
XSL,
XSLT を参照, 895
XSL アウトライン, 325
XSL アウトラインウィンドウ, 326
XSL 概要ウィンドウ, 109
XSL 変換,
XSLT を参照, 94
XSL/XQuery メニュー, 886
xsl:call-template, 326
XSL:FO,
と XSLT 変換, 323
xsl:param, 326
xsl:with-param, 326
XSLT, 969
XMLSpy での変更, 96
XMLSpy における機能, 320
XMLSpy を使った変換, 323
テキストビューにおける自動補完, 321
ドキュメント, 321
とバッチ変換, 323
プロセッサ, 1036
拡張関数, 1561
検証, 321
入力ヘルパー, 321
XSLT スタイルシート,
XML ドキュメントの割り当て, 894
XML ドキュメントへ割り当て, 894
開く, 895
XSLT デバッガー,
Visual Studio .NET 内の XSLT デバッガー, 733
ステップアウト, 896
ステップイン, 896
ステップオーバー, 897
デバッガーを開始, 895
デバッガーを再起動, 896
デバッガーを停止, 895
デバッグウィンドウ, 899
デバッグセッションを終了, 896
トレースポイントの挿入/削除, 897
トレースポイントの有効化/無効化, 898
ブレークポイント/トレースポイントダイアログボックス, 898
ブレークポイントの挿入/削除, 897
ブレークポイントの有効化/無効化, 898
現在実行中のノードを表示, 897
設定, 899
XSLT テンプレート,
XSL アウトラインウィンドウにて, 326
XSLT と XQuery のためのJava 拡張関数,
Java 拡張関数を参照する, 1561
XSLT と XQuery のための拡張関数, 1561
XSLT ドキュメント,
XSL アウトラインを使った編集と管理, 325
と情報ウィンドウ, 329
XSLT ドキュメント構造, 109
XSLT パラメーター,
インターフェイスを使用してスタイルシートにパスする, 890
XSLT プロセッサー,
Altova 製品内で, 1591
XSLT へ変換,
XSLT ファイルを割り当てる, 94
XSLT 関数,
XSL アウトラインウィンドウにて, 326
XSLT 変換, 888
FO への変換, 888
PDF への変換, 888
XMLSpy 内で, 95
チュートリアル, 94
XSLT/XQuery デバッガー, 357
XPath-Watch ウィンドウ, 367
インターフェースの記述, 357
キーボードショートカット, 380
コーススタックウィンドウ, 367
コンテキストウィンドウ, 365
セッションの開始, 363
ツールバーアイコン, 358
テンプレートウィンドウ, 369

XSLT/XQuery デバッガー, 357
トレースウィンドウ, 370
トレースポイントウィンドウ, 375
ブレークポイントの使用方法, 372
メカニズムの記述, 357
メッセージウィンドウ, 368
機能と使用方法, 356
情報ウィンドウ, 364, 369
設定, 362
変数ウィンドウ, 366
XSLT/XQuery デバッグ,
使用するファイル, 356
XSLT/XQuery 内のスクリプト,
拡張関数で参照する, 1561
XSLT/XQueryを使用した JSON 変換, 541

Z

ZIP ファイル, 225, 560
Archive View にて作成, 566
Archive View にて編集, 566
アイコン,
ツールバー/メニューへ追加, 998
大きく表示, 1011
アクティブな構成,
グローバルリソース, 994
アサーションメッセージ, 212
アップデートリストの保留 (PUL), 345
イメージ書式設定,
内の Authentic View, 450
インターネット, 1052
インターネットの使用,
Altova 製品内で, 1592
インテリジェントな編集, 1021
インデント,
テキストビューにて, 771
インデントガイド, 969
インポート,
MS Word ドキュメントを XML として, 946
XML スキーマをベースにデータベースデータをインポート, 950
データベースデータを XML として, 941
データベース構造を XML スキーマとして, 946
テキストファイルから XML へ, 938
ウィンドウ,
フロート、ドッキング、タブ, 103
プロジェクト, 1044, 1045
開く, 1046
左右に並べて表示, 1044
自動非表示, 103
重ねて表示, 1044
上下に並べて表示, 1044
情報, 1045
入力ヘルパー, 1045
ウィンドウ メニュー, 1044
エイリアス,
グローバルリソース参照, 660
エクスプローラー, 1018
エンコーディング,
デフォルト, 1020
ファイル, 751
エンティティ,
Authentic View に挿入, 412
Authentic View 内に挿入, 428
Authentic View 内の定義, 428
XML スキーマをベースにする XML にて, 232
内の Authentic View の定義, 448
エンティティ入力ヘルパー,
Authentic View, 421
エンドユーザー使用許諾契約書, 1593, 1594
オーバーライド, 161
オープン コンテンツ モデル, 185
オブジェクトロケーター,
データベースクエリウィンドウにて, 917
オプションの要素,
にする, 52
お気に入り, 1047
カスタマイズ, 114
コンテキストメニュー, 1006
コンテキストメニューのカスタマイズ, 1011
ツールバー/メニューコマンド, 998
マクロ, 1008
メニュー, 1006
カスタムカタログ, 841
カスタム辞書, 974
カタログ, 301
Oasis XML, 841
カタログとインテリジェントな編集, 303
カタログと環境変数, 305
カタログのカスタム化, 303
カタログメカニズムの概要, 301
カタログ内で使用される環境変数, 302
キーコード,
Altova ソフトウェアのための, 1049
キーボードショートカット, 1003

キーマップ, 1048
クエリ,
Authentic View内の DB 表示, 441
XQuery を参照, 912
データベースクエリウィンドウ以下を参照, 912
クエリデータベース ウィンドウ, 912
クエリデータベースコマンド, 912
クエリペイン,
データベースクエリウィンドウにて, 920
グリッドビュー, 87, 141, 966, 967
データの入力, 78
テーブル, 143
テーブルビューに切り替える, 837
とデータのインポート/エクスポート, 143
とテーブルビュー, 89
入力ヘルパー, 147
入力ヘルパーの使用, 87
編集, 141, 838
要素と属性の付加, 87
グリッドビュー (JSON),
JSON グリッドビューを参照, 475
グリッドビュー (JSON) テーブル,
TSV または JSON としてコピー, 502
グリッドビュー (JSON) 設定, 502
グリッドビュー (XML), 236
XQuery 式, 254, 255
コンテキストメニュー, 262
チャートの作成, 259
テーブル表示アイコン, 250
ドキュメントコンテンツの編集, 243
ドキュメントの表示, 238
ドキュメント構造の編集, 241
ドラッグアンドドロップ, 246
ノードの型を編集する, 243
ノードの追加と削除, 241
メタデータファイル, 254, 255
機能, 238
検索と置換, 243
検証, 243
出力を計算するためのフォーミュラ, 254
内でのイメージ, 258
入力ヘルパー, 241
表示のフィルター, 255
グリッドビュー (XML) テーブル,
TSV または XML としてコピー, 265
グリッドビュー (XML) 設定, 265
グリッドビューでの XML のための入力ヘルパー, 249
グリッドビュー内のイメージ, 258
グリッドビュー内のチャート, 259
グリッドビュー内の行,
挿入, 837
付加, 837
グリッドビュー入力ヘルパー, 249
グリッドフォント, 1031
グレーのバー, 967
グローバル,
設定, 1014
グローバル リソース,
定義, 661
グローバル リソース XML ファイル, 661
グローバルコンポーネント,
グローバルコンポーネントの名前変更, 317
グローバルなコメント,
表示される行, 157
グローバルリソース,
アクティブな構成, 994
ツールバーにて有効にする, 1000
データベースの型の定義, 668
ファイルの型の定義, 663
ファイル型とフォルダー型の使用, 670
フォルダーの型の定義, 667
構成の変更, 674
使用, 670, 673, 674
定義, 993
グローバルリソースの,
構成, 661
グローバルリソースの構成, 674
グローバル属性, 161
グローバル属性グループ, 161
グローバル要素, 161
XML スキーマを使用する, 63
コーススタックウィンドウ,
XSLT/XQuery デバッガー, 367
コードページ, 1031
コールスタック (XPath/XQuery), 389
コピー,
XML テキストとして, 764
構造化テキストとしての XML, 764
コピーコマンド, 763
コマンド, 998
キーマップにて表示, 1048
コンテキストメニュー, 1006
ツールバー/メニューへ追加, 998
メニューから削除, 1006
メニューのリセット, 1006
コマンドライン, 1053

コメント, 157
グリッドビューにて子として追加, 829
グリッドビューにて挿入, 814
グリッドビューにて追加, 822
グリッドビューにて変換, 835
コメントインとアウト,
テキストビュー内の XML ドキュメント, 234
コンテキストウィンドウ,
XSLT/XQuery デバッガー, 365
コンテキストメニュー,
カスタマイズ, 1011
コマンド, 1006
コンテンツ モデル ビュー,
インターフェイスの詳細, 170
コンポジッターとコンポーネント, 170
一般的な説明, 168
条件型のい割り当て, 181
図に表示されるオブジェクト, 170
編集, 176
コンテンツモデル,
ダイアグラムの保存, 868
基本的なモデルの作成, 44
切り替え属性, 65
コンテンツモデルビュー, 41
設定, 873
コンポーネント定義,
再利用する, 56
コンポーネント入力ヘルパー, 201, 207
コンポジッター,
シーケンスコンポジッター, 44
サーチ,
検索を参照, 776
サポートオプション, 117
サポートセンター, 1052
シーケンスコンポジッター,
使用する, 44
ショートカット, 227
ツールチップにて表示, 1011
割り当て/削除, 1003
ズーム機能,
スキーマデザインビューにて使用, 876
スキーマ,
DB XML へ割り当て, 929
IBM DB2 データベースにおける管理と割り当て, 925
IBM DB2 にて管理, 925
UML へ変換, 859
XML スキーマも参照, 852
XML スキーマを参照, 41
カタログを介したルックアップ, 303
スキーマをベースに DB を作成, 951
デザインビュー, 966
ドキュメンテーション, 70
ドキュメント フォント, 1034
メモリ内, 864
設定, 1015
スキーマ コンポーネント,
のコンテンツ モデル, 168
スキーマ ビュー,
コンポーネント入力ヘルパー, 201, 207
詳細入力ヘルパー, 205
入力ヘルパー, 201
戻る、進む, 222
スキーマ ビューにおけるアサーション, 187, 191
スキーマ ビューにおけるコメント, 161
スキーマ ビューにおけるコンポジッター, 170
スキーマ ビューにおける属性, 187, 188
スキーマ ビューにおける属性グループ, 188
スキーマ ビューにおける注釈, 161
スキーマ ビューにおける同一性制約, 187, 194
スキーマ ビューにおける入力ヘルパー, 201
スキーマ フォント, 1034
スキーマサブセット, 285, 883, 884
スキーマデザインビュー,
ズーム機能, 876
ダイアグラムの表示, 877
全てのグローバルを表示, 877
スキーマデザインメニュー, 865
スキーマとカタログ, 301
スキーマの修正,
コンテンツ モデル, 168
スキーマの中を検索,
ウィンドウ, 317
スキーマの中を検索, 307
結果, 317
範囲の設定, 314
スキーマの中を検索ウィンドウ, 112
スキーマの中を検索を参照,
スキーマの中を検索, 307
スキーマビュー, 151
ビューの構成, 50
スキーマビュー, 検索,
スキーマの中を検索, 307
スキーマビュー内での命令の処理, 161
スキーマビュー内の注釈, 161
スキーマ概要, 41
アイコン, 157

スキーマ概要, 41
グローバルコメントのライン表示, 157
グローバルなコメント, 157
コンポーネントの並べ替え, 157
とコンテンツ モデル ビュー, 156, 157
編集, 157
スキーマ型の生成, 215
スキーマ内の検索, 307
スキーマの中を検索, 317
プロパティとプロパティの値による検索結果の絞り込み, 311
検索と名前変更, 317
検索ならびに置き換えコマンドの実行, 315
検索をコンポーネントに絞り込む, 310
検索語, 309
置き換え語, 309
スクリプト, 1040
スクリプトエディター,
開始, 982
スクリプト言語, 1040
スタートグループ,
追加(コンテキストメニュー), 1011
スタイル, 1031
スタイルシート処理命令, 894
ステータスバー, 112
ステップアウト,
XSLT デバッガー, 896
ステップイン,
XSLT デバッガー, 896
ステップオーバー,
XSLT デバッガー, 897
スプラッシュ画面, 1025
スペルチェッカー,
カスタム辞書, 974
スペルチェックのオプション, 978
スマート制約, 215
セルの最大幅, 1025
ソースコントロール,
チェックアウトを元に戻す, 791
プロバイダーの変更, 798
ソース管理, 1040
サポートされているプロバイダー, 785
ステータスを最新の状態に更新, 798
ソース管理に追加, 791
チェックアウト, 789
ファイルの削除, 792
フォルダーの取得, 788
プロジェクトを開く, 786
プロパティ, 797
共有, 792
差分の表示, 795
最新バージョンの取得, 788
有効化、無効化, 787
履歴の表示, 794
ソース管理マネジャー, 798
ソフトウェアのライセンスの認証, 1049
ソフトウェア製品ライセンス, 1594
ダイアグラムの表示, 877
タブのサイズ,
および整形, 969
設定, 969
タブ文字, 1015
チェック,
スペルチェッカー, 974
ツール,
外部アプリケーションを参照, 1002
ツールチップ,
ショートカットを表示, 1011
表示, 1011
ツールバー, 112
コマンドを追加, 998
ツールバーならびにメニューコマンドのリセット, 1000
マクロの追加, 1008
新規作成, 1000
大きなアイコンで表示, 1011
有効化/無効化, 1000
ツールメニュー, 974
ディレクトリ,
2つのディレクトリを比較, 987
ディレクトリの比較, 701, 703, 987
データのインポート/エクスポート,
とグリッドビュー, 143
データベース,
Authentic Viewにて編集, 902
DB を参照, 440
Oracle XML Db, 933
XML データをエクスポート, 957
XMLSpy 内でのサポート, 659
スキーマをベースに DB を作成, 951
データを XML としてインポート, 941
とグローバルリソース, 673
レコードの編集, 923
構造を XML スキーマにインポート, 946
データベース クエリ,
メッセージペイン, 923
結果の編集, 923

データベース/テーブルビュー,
使用方法, 89
データベースクエリ,
DB クエリウィンドウ内のブラウザーペイン, 917
DB へ接続して問い合わせ, 914
クエリの作成, 922
データベースドライバー,
概要, 575
データベースレコードの編集, 923
データベース接続,
ウィザードの開始, 574
グローバルリソースからの再使用, 602
設定, 573
テーブル,
Authentic View 内, 407
グリッドビューのテーブル, 143
自動的に生成, 1018
動的な(SPS) テーブルの編集, 416
テーブル 内の Authentic View,
SPS (静的と動的) テーブルの使用, 432
XML テーブルの編集のためのアイコン, 437
テーブルコマンド,
グリッドビュー内, 836
テーブルビュー, 1021
グリッドビューに切り替える, 837
行の挿入, 837
行の付加, 837
使用方法, 89
列の並べ替え, 837, 838
テーブル内の Authentic View,
XML テーブルの使用, 433
テキスト,
Authentic View 内の書式設定, 428
Authentic View 内の編集, 428
グリッドビューにて子として追加, 829
グリッドビューにて挿入, 814
グリッドビューにて追加, 822
グリッドビューにて変換, 835
ドキュメント内を検索, 773
検索と置き換え, 776
整形, 771
テキストの範囲,
内の Authentic View, 425
テキストビュー, 122, 965
XML ドキュメントのための特別な編集機能, 234
XML ドキュメント内のコメント, 234
インテリジェントな編集機能, 127
インデント, 123, 124
キーボードショートカット, 140
スキーマ フォント, 1034
ソース折りたたみ, 124
テキストのフォーマット, 123
と XML ドキュメント内の空のライン, 234
フォントプロパティ, 123
ブックマーク, 124
ワードラップ, 123
行番号, 124
自動補完の有効化/無効化, 1021
内の分割機能, 138
入力ヘルパー, 137
編集, 79
テキストビューにおけるソースの折りたたみ, 124
テキストビューにおけるブックマーク, 124
テキストビューにおける行番号, 124
テキストビューにおける入力ヘルパー, 137
テキストビューにおける編集, 127
テキストビューのズーム, 124
テキストビュー設定ダイアログ, 969
テキストビュー内でのインデント, 123
テキストビュー内でのフォーマット, 123
テキストビュー内でのフォント, 123
テキストビュー内でのワードラップ, 123
テキストビュー内の分割ビュー, 138
テキストファイル,
XML データからエクスポート, 954
XML としてインポート, 938
テクニカルサポート, 1052
デバッガー,
XSLT デバッガーを開始, 895
XSLT デバッガーを再起動, 896
XSLT デバッガーを停止, 895
ステップアウト, 896
ステップイン, 896
ステップオーバー, 897
セッションを終了, 896
デバッグウィンドウ, 899
トレースポイントの挿入/削除, 897
トレースポイントの有効化/無効化, 898
ブレークポイント/トレースポイントダイアログボックス, 898
ブレークポイントの挿入/削除, 897
ブレークポイントの有効化/無効化, 898
現在実行中のノードを表示, 897
設定, 899
デバッガーウィンドウ,
配置, 370
デフォルト,

デフォルト,
エンコーディング, 1020
メニュー, 1006
デフォルト オープン コンテンツ モデル, 185
デフォルトエディター, 1018
デフォルトビュー,
メインウィンドウにおける設定, 1018
テンプレート,
Authentic View における XML ドキュメントのテンプレート, 901
テンプレート XML ファイル,
Authentic View 内で開く, 404
テンプレートウィンドウ,
XSLT/XQuery デバッガー, 369
テンプレートファイル,
新規ドキュメントに対して, 742
ドキュメンテーション,
XML スキーマファイルから生成, 868
スキーマ, 70
ドキュメント,
Oracle XML Db の参照, 936
スペルチェッカー, 974
ドキュメントレベル,
XMLSpy の統合のサンプル, 1421
ドッキングウィンドウ, 103
ドック可能なウィンドウ, 1044, 1045
トピック,
目次にて参照, 1047
トレースウィンドウ, 375
XSLT/XQuery デバッガー, 370
トレースポイント,
XSLT/XQuery デバッガー, 375
ダイアログボックス, 898
トレースポイントの有効化 - XSLT デバッガー, 898
ナビゲーション,
スキーマデザインでのショートカット, 68
ノード,
現在実行中のノードを表示, 897
パーサー,
Altova 製品に内蔵の, 1591
バックアップ, 120
パラメーター,
DB クエリ内, 441
インターフェイスを使用してスタイルシートにパスする, 890
バリデーター,
Altova 製品内で, 1591
ビッグエンディアン, 1020
ビュー,
スキーマデザインビュー, 966
テキストビュー, 965
ファイルへ移動, 969
ブラウザー ビュー, 967
行/文字へ移動, 969
最適な幅, 968
縮退, 968
縮退一, 967
展開, 967, 968
ビューの変更,
Authentic View, 416
ビューメニュー, 965
ビルダーモード, 395
ファイル, 1015
2つを比較, 983
エンコーディング, 751
ソース管理に追加, 791
タブ, 1015
デフォルトエンコーディング, 1020
メールで送信, 757
印刷オプション, 758
開く, 746
開くオプション, 1015
最後に使用, 760
新たに作成, 742
比較オプション, 990
閉じる, 752
保存, 752
ファイル DSN,
セットアップ, 591
ファイル コマンド内の検索, 778
ファイル メニュー, 742
ファイルタイプ, 1018
ファイルの中を検索ウィンドウ, 110
ファイルの比較, 701, 702
オプション, 990
ファイルの保存,
エンコーディング, 751
ファイルパス,
XML ドキュメントの挿入, 234
ファイルへ移動, 969
ファイル拡張子,
XQuery ファイル, 334
カスタマイズ, 841
ファイル比較の設定, 990
フィルターとフォーミュラのための JSON メタデータ, 492
フィルターのための XML メタデータファイル, 254
フォーミュラのための XML メタデータファイル, 255

フォント, 1034
スキーマ, 1034
スキーマ ドキュメント, 1034
フォントサイズ,
ブラウザービューにて, 973
ブックマーク, 1047
ナビゲーション, 780
挿入と削除, 780
ブックマークの余白, 969
ブラウザー, 1025
ビュー, 967
ブラウザー ビュー, 972
ブラウザー メニュー, 972
ブラウザービュー,
フォントサイズ, 973
ページの読み込みを中止, 973
最新の状態に更新, 973
進む, 972
別のウィンドウ, 973
戻る, 972
ブラウザーペイン,
データベースクエリウィンドウにて, 917
プラットフォーム,
Altova 製品のための, 1591
ブレークポイント,
XSLT/XQuery デバッガーで使用する, 372
ダイアログボックス, 898
ブレークポイントダイアログ, 358
ブレークポイントの有効化 - XSLT デバッガー, 898
プログラマーのレファレンス, 1054
プログラム設定, 1014
プロジェクト, 800
URL の追加, 799
アクティブなファイルを追加, 799, 800
グローバルリソースの追加, 799
ソース管理に追加, 791
ファイルの追加, 799
フォルダーの追加, 800
プロパティ, 677, 808
を使ったバッチ処理, 681
開く, 784
外部ウェブフォルダーの追加, 803
外部フォルダーの追加, 800
概要, 676, 782
関連するファイルを追加, 800
再ロード, 785
最近使用されたプロジェクト, 811
作成と編集の方法, 677
使用する, 681
使用する利点, 681
新規作成, 784
閉じる, 785
保存, 677, 785
名前をつける, 677
プロジェクト ウィンドウ, 105
プロジェクトウィンドウ, 1044, 1045
表示/非表示の切り替え, 1045
プロジェクトメニュー, 782
ヘルプ,
キーマップ, 1048
目次, 1047
ヘルプ メニュー, 1047
ヘルプシステム, 1047
ホットキー, 1003
マークアップ,
内の Authentic View, 416, 419
マークアップ（Authentic View にて）,
隠す, 908
小さな/大きな/混合マークアップを表示, 908
マークアップの非表示, 416
マークアップを隠す, 419
マークアップを隠す（Authentic View にて）, 908
マクロ,
アプリケーションマクロの起動, 983
メニュー/ツールバーへ追加, 1008
編集ボタン, 1011
メインウィンドウ, 104
メインウィンドウにおけるドキュメント, 104
メインカタログ, 841
メタデータファイル, 254, 255, 492
メッセージウィンドウ,
XSLT/XQuery デバッガー, 368
メニュー, 998
Authentic, 901
DTD/スキーマ, 852
JSON, 848
XML, 812
XSL/XQuery, 886
ウィンドウ, 1044
カスタマイズ, 1006
コマンドの追加/削除, 998
コマンドを削除, 1006
スキーマデザイン, 865
デフォルト/XMLSpy, 1006
ビュー, 965
プロジェクト, 782

メニュー, 998
ヘルプ, 1047
マクロの追加, 1008
ルーツ, 974
変換, 938
編集, 762
メニュー ブラウザー, 972
メニューバー, 112
メモリ,
スキーマ情報の保管, 864
メモリ要件, 1591
モデルグループ, 161
やり直しコマンド, 763
ユーザー DSN,
セットアップ, 591
ユーザーインターフェイス詳細, 103
ユーザーマニュアル, 22
ユーザーリファレンス, 741
ライセンス, 1594
Altova ソフトウェアのための, 1049
情報, 1593
ライセンス計測,
Altova 製品にて, 1593
リセット,
ショートカット, 1003
ツールバーならびにメニューコマンド, 1000
メニューコマンド, 1006
リッチエディット, 908
リトルエンディアン, 1020
リンク,
Authentic View 内の, 428
レジストリ,
設定, 1014
ワードラップ,
有効化/無効化, 968
位置,
行, 969
文字, 969
移動ウィンドウ, 103
印刷,
Authentic View から, 413
印刷オプション, 758
印刷設定, 760
隠す, 1044, 1045
右へ移動コマンド, 838
解析,
XSLT, 1036
改行, 1015
改行マーカー, 969
開く,
ファイル, 746
開くオプション,
ファイル, 1015
開始,
XSLT デバッガーを開始, 895
外部 JSON コンテンツ,
JSON グリッドビューにコピーする, 485
JSON グリッドビューにドラッグアンドドロップする, 485
外部 XML コンテンツ,
XML グリッドビューにコピーする, 246
XML グリッドビューにドラッグアンドドロップする, 246
外部 XSL プロセッサ, 1036
外部アプリケーション,
ファイルから開く, 1002
外部キー,
SQLite 内での無効化, 601
外部ソースからの JSON テキスト,
ドキュメントに素早く追加, 539
外部ソースからの XML テキスト,
ドキュメントに素早く追加, 273
外部で解析されるエンティティ, 1018
拡張グリッドビュー, 966
グリッドビューを参照, 78
拡張検証, 298
SchemaAgent にて実行, 882
割り当て,
ショートカットをコマンドへ割り当て, 1003
環境変数, 305
基底型,
修正, 214
技術データ, 1591
空のライン,
テキストビュー内の XML ドキュメント, 234
空白,
削除, 772
空白の削除, 772
空白マーカー, 969
空要素, 1018
繰り返される要素, 1021
型,
XML ドキュメントでの拡張, 76
検索,
ドキュメント内のテキスト, 773
と置き換え(ドキュメント内のテキストにて), 776
正規表現の使用, 132
検証, 114, 841

検証, 114, 841
DTD を XML ドキュメントへ割り当て, 852, 853
SchemaAgent により関連するスキーマの検証, 298
WSDL ファイル, 846
XML ドキュメント, 83
検証の設定, 1015
元に戻すコマンド, 763
現在実行中のノードを表示,
XSLT デバッガー, 897
構造化されたテキスト, 1021
構文による色分け, 336, 1018, 1025
XQuery, 336
行,
移動, 969
削除(Authentic View にて), 909
上/下へ移動, 910
挿入(Authentic View にて), 909
追加(Authentic View にて), 909
複製(Authentic View にて), 909
行/文字へ移動, 969
行番号の余白, 969
混合マークアップ(Authentic View にて), 908
左へ移動コマンド, 838
再ロード, 1015
ファイルの変更, 751
再起動,
XSLT デバッガーを再起動, 896
再定義, 161
最後に使用されたファイル,
リスト, 760
最適な幅, 968, 1025
作成,
スキーマをベースにした DB, 951
削除,
コンテキストメニューのコマンドを削除, 1006
ショートカット, 1003
ツールバー, 1000
ツールバーからアイコンを削除, 998
ツールバーからコマンドを削除, 998
トレースポイント - XSLT デバッガー, 897
ブレークポイント - XSLT デバッガー, 897
行(Authentic View にて), 909
削除コマンド, 763
参照,
Oracle XML Db, 936
子の追加コマンド,
グリッドビューにて, 827
視覚的な書式設定,
内の Authentic View, 450
試用期間,
Altova ソフトウェア製品の試用, 1593
自動バックアップの設定, 1015
自動検証, 1018
自動検証の無効化, 1018
自動的なバックアップ, 120
自動非表示ウィンドウ, 103
自動補完,
テキストビューにて有効化/無効化, 1021
辞書,
スペルチェッカー, 974
既存のものを修正, 974
新たに追加, 974
式ビルダー, 395
終了,
デバッガーセッションを終了, 896
重ねて表示,
ウィンドウ, 1044
縮退,
未選択を縮退, 968
出現,
回数, 44
出力ウィンドウ,
表示/非表示の切り替え, 1045
出力フォーマット, 1015
処理命令,
グリッドビューにて子として追加, 830
グリッドビューにて挿入, 815
グリッドビューにて追加, 823
グリッドビューにて変換, 835
小さなマークアップ, 419
小さなマークアップ (Authentic View), 908
小さなマークアップの表示, 416
章, 1047
詳細入力ヘルパー, 44, 205
上/下へ移動,
行(Authentic View にて), 910
情報 ウィンドウ, 1045
情報ウィンドウ, 107
CSS ドキュメントに対して, 457
HTML ドキュメントに対して, 455
XSLT/XQuery デバッガー, 369
配置, 370
情報ウィンドウ XSLT タブ,
XSL アウトラインも参照, 329
と XSLT ドキュメント, 329
と ZIP フォルダーの作成, 329

情報ウィンドウ XSLT タブ,
とプロジェクト, 329
の記述, 329
条件型の割り当て, 181
条件型の代替, 181
色, 1031
新規の XML ドキュメントのため,
作成, 74
新規ファイル,
作成, 742
新規機能, 23, 24
進む,
スキーマ ビューにて, 222
整形,
テキストビューにて, 771
整形式チェック,
XML ドキュメントのため, 83
整形式のチェック, 839
正規表現, 778
検索と置換の条件式, 132
検索文字列にて, 773
生成,
スキーマをベースに DB 構造を生成, 951
静的な (SPS) テーブル 内の Authentic View,
使用方法, 432
切り取りコマンド, 763
切り替え, 1044, 1045
折りたたみ余白, 969
折り返し,
ワードラップの有効化/無効化, 968
設定, 114, 1014
XSLT デバッガー, 899
スクリプト, 1040
の概要, 38
設定ビュー,
コンテンツモデルビューのダイアログ, 873
選択されたノードに対しての XPath, 415
全てのグローバルを表示, 877
全て選択コマンド, 772
挿入, 909
トレースポイント - XSLT デバッガー, 897
ブレークポイント - XSLT デバッガー, 897
属性, 65, 161
グリッドビューにて子として追加, 828
グリッドビューにて挿入, 813
グリッドビューにて追加, 821
グリッドビューにて変換, 835
コンテンツモデルビューに切り替え, 65
スキーマ定義にて, 65
属性のプレビュー, 1025
属性値,
Authentic View 内に挿入, 411
属性入力ヘルパー,
Authentic View, 421
大きなマークアップ, 419
大きなマークアップ(Authentic View にて), 908
大きなマークアップの表示, 416
単純型,
アサーション, 207
グローバル, 161
サンプル値, 207
スキーマ定義にて, 56
パターン, 207
ファセット, 207
匿名, 161
名前を持つ, 161
列挙, 207
単純型のアサーション, 207
単純型のサンプル値, 207
単純型のパターン, 207
単純型のファセット, 207
単純型の派生, 205
単純型の列挙, 207
置き換え, 110
置換,
正規表現の使用, 132
置換え,
テキスト, 773
ドキュメント内のテキスト, 776
複数のファイル内のテキスト, 778
著作権に関する情報, 1593
追加コマンド,
グリッドビューにて, 820
停止,
XSLT デバッガーを停止, 895
定義概要グリッド, 505
適用, 1014
展開,
全て展開, 968
貼り付け,
XML, 425
XML として, 428
テキスト, 425
テキストとして, 428
貼付けコマンド, 763
電子メール,

電子メール,
ファイルを送信, 757
統合,
アプリケーション内の XMLSpy, 1413
動的な (SPS) テーブル 内の Authentic View,
使用方法, 432
動的なテーブル,
編集, 416
同一性制約,
コンテンツモデルビューに切り替え, 65
内の Authentic View テーブル,
使用方法, 432
日付,
手動での変更, 447
日付の選択,
内の Authentic View の使用, 446
入力ヘルパー, 107, 1045
XQuery, 335
グリッドビュー内, 87
更新, 847
詳細, 44
表示/非表示の切り替え, 1045
入力ヘルパー (テキストビュー、Authentic ビュー), 270
入力ヘルパーの更新コマンド, 847
派生型,
基底型の切り替え, 214
背景情報, 1591
配置,
左右に並べて, 1044
上下に並べて, 1044
配布,
Altova ソフトウェア製品, 1593
Altova ソフトウェア製品の配布, 1593
比較,
ディレクトリの比較, 703
ファイルとディレクトリの比較, 701
ファイルの比較, 702
非 XML ファイル, 1018
表示, 1044, 1045
拡張グリッドビュー, 966
表示方法, 1025
評価キー,
Altova ソフトウェアのための, 1049
評価機関,
Altova ソフトウェア製品, 1593
複合型, 56
グローバル, 161
スキーマ定義にて, 56
定義を拡張する, 56
匿名, 161
名前を持つ, 161
複合型マークアップ, 419
複合型マークアップの表示, 416
複数ユーザー, 1015
文字,
位置, 969
文字セット,
エンコーディング, 1020
文法, 1018
変換,
MS Word データから XML へ, 946
Oracle XML Db, 933
XSLT 変換を参照, 888
スキーマから DB 構造へ, 951
データベーススキーマを XML スキーマへ, 946
データベースデータを XML として, 941
テキストファイルから XML へ, 938
変換 メニュー, 938
変換コマンド,
グリッドビューにて, 834
変更コマンドでの検証, 847
変更を検知, 1015
変数ウィンドウ,
XSLT/XQuery デバッガー, 366
編集,
マクロボタン, 1011
編集ビュー, 118
編集メニュー, 762
法的な情報, 1593
未選択, 968
無効化,
トレースポイント - XSLT デバッガー, 898
ブレークポイント - XSLT デバッガー, 898
名前なし要素リレーションシップ,
MS SQL Server スキーマ設定, 880
名前空間,
スキーマデザインビューにお, 865
スキーマ内の, 43
名前空間プレフィックス,
グリッドビューにて追加, 847
名前付きスキーマのリレーションシップ,
MS SQL Server スキーマ設定, 880
名前付きテンプレート, 326
名前付きテンプレートの呼び出し, 326
目次, 1047
戻る,

戻る,
スキーマ ビューにて, 222
要素, 52
オプションにする, 52
グリッドビューにて子として追加, 828
グリッドビューにて挿入, 813
グリッドビューにて追加, 821
グリッドビューにて変換, 835
コンテンツの制限, 52
要素の型,
XML ドキュメントで指定する, 76
要素の中へコマンド, 838
要素入力ヘルパー,
Authentic View, 421
履歴, 222
隣り合わせ, 1025
列挙,
XMLSpyControl, 1479
属性を定義する, 65