Benutzerhandbuch und Referenz für Programmierer

Copyright © 1998-2007. Altova GmbH. All rights reserved. Use of this software is governed by and subject to an Altova software license agreement. XMLSpy, MapForce, StyleVision, SemanticWorks, SchemaAgent, UModel, DatabaseSpy, DiffDog, Authentic, AltovaXML, MissionKit, and ALTOVA as well as their logos are trademarks and/or registered trademarks of Altova GmbH.

ALTOVA®

XML, XSL, XHTML, and W3C are trademarks (registered in numerous countries) of the World Wide Web Consortium; marks of the W3C are registered and held by its host institutions, MIT, INRIA, and Keio. UNICODE and the Unicode Logo are trademarks of Unicode Inc. This software contains 3rd party copyrighted software or material that is protected by copyright and subject to other terms and conditions as detailed on the Altova website at http://www.altova.com/legal_3rdparty.html

Altova XMLSpy 2007 Enterprise Edition Benutzer- und Referenzhandbuch

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.

Published: 2007

© 2007 Altova GmbH

Inhaltsverzeichnis

Willkommen bei der XMLSpy Enterprise Edition 3

Benutzerhandbuch 6

1 Einführung 7

1.1 Die XML-Produkte von Altova 8
1.2 Die wichtigsten Funktionen von XMLSpy 10
1.3 Benutzeroberfläche 14
1.3.1 Projektfenster 15
1.3.2 Info-Fenster 16
1.3.3 Hauptfenster 16
1.3.4 Eingabehilfen 17
1.3.5 Menü- und Symbolleiste 18
1.3.6 Ausgabefenster 19

2 Tutorials 20

2.1 XMLSpy Tutorial 21
2.1.1 Die XMLSpy-Benutzeroberfläche 21
2.1.2 Erstellen eines einfachen XML-Schemas 22
– Anlegen einer neuen XML-Schema-Datei 23
– Definieren von Namespaces 26
– Definieren eines Content Model 27
– Hinzufügen von Elementen mit Drag & Drop 31
– Konfigurieren der Content Model-Ansicht 32
– Fertigstellen des Schemas 34
2.1.3 Komplexere XML-Schema-Definitionen 37
– Arbeiten mit ComplexTypes und SimpleTypes 38
– Referenzieren von globalen Elementen 45
– Attribute und Attribut-Enumerationen 47
2.1.4 Schemanavigation und Dokumentation 49
– Schemanavigation 50
– Schema-Dokumentation 52

2.1.5 Erstellen von XML Dokumenten ... 55
– Erstellen eines neuen XML-Dokuments ... 56
– Festlegen des Typs eines Elements ... 58
– Eingabe von Daten in der Grid-Ansicht ... 60
– Eingabe von Daten in der Textansicht ... 60
– Validierung des Dokuments ... 64
– Anhängen von Elementen und Attributen in der Grid-Ansicht ... 68
– Editieren in der Datenbank/Tabellen-Ansicht ... 70
– Ändern des Schemas ... 74
2.1.6 Transformieren von XML mit XSLT ... 76
– Zuweisen einer XSL-Datei ... 77
– Transformieren der XML-Datei ... 77
– Ändern der XSL-Datei ... 78
2.1.7 Projektverwaltung ... 80
– Vorteile von Projekten ... 80
– Erstellen eines Projekts ... 81
2.1.8 Fertig! ... 82
2.2 Tutorial Authentic-Ansicht ... 83
2.2.1 Öffnen eines XML-Dokuments in der Authentic-Ansicht ... 84
2.2.2 Die Benutzeroberfläche der Authentic-Ansicht ... 85
2.2.3 Node-Operationen ... 87
2.2.4 Eingabe von Daten in der Authentic-Ansicht ... 90
2.2.5 Eingabe von Attributwerten ... 93
2.2.6 Hinzufügen von Entities ... 93
2.2.7 Drucken des Dokuments ... 94
2.3 WSDL Tutorial ... 96
2.3.1 Erstellen eines neuen Dokuments ... 96
2.3.2 Erstellen eines PortType ... 97
2.3.3 Erstellen eines Binding ... 99
2.3.4 Erstellen eines Service und von Ports ... 100
2.3.5 Validieren des WSDL-Dokuments ... 101
2.3.6 Herstellen einer Verbindung zu einem Webservice und Öffnen von Dateien ... 101
2.3.7 Senden eines SOAP Request von der WSDL-Datei ... 102
2.3.8 Erstellen von WSDL-Dokumentation ... 103

3 Textansicht 105

3.1 Visuelle Editierhilfen in der Textansicht ... 106
3.2 Eingabehilfen in der Textansicht ... 107
3.3 Ausgabefenster ... 109
3.3.1 Validierung ... 110
3.3.2 In Dateien suchen ... 111
3.3.3 XPath ... 112
3.4 Bearbeiten von XML-Dokumenten ... 117

3.5 Bearbeiten von XQuery-Dokumenten 120
3.5.1 Öffnen eines XQuery-Dokuments 120
3.5.2 XQuery Eingabehilfen 121
3.5.3 XQuery Syntaxfärbung 122
3.5.4 Intelligentes Editieren von XQuery 124
3.5.5 Validierung und Ausführung von XQuery-Dokumenten 125
3.5.6 XQuery und XML-Datenbanken 126
3.6 Bearbeiten von CSS-Dokumenten 131

4 Enhanced Grid-Ansicht 134

4.1 Editieren in der Grid-Ansicht 138
4.2 Datenbank/Tabellen-Ansicht 140

5 Schema/WSDL Design-Ansicht 144

5.1 Schema Design-Ansicht 145
5.1.1 Schema-Übersicht 150
5.1.2 Content Model-Ansicht 154
5.1.3 Smart Restrictions 164
5.1.4 Arbeiten mit SchemaAgent 168
– Herstellen einer Verbindung mit SchemaAgent Server 170
– Öffnen von Schemas aus dem Suchpfad 172
– Verwenden von Schema-Konstrukten 173
– Anzeigen von Schemas in SchemaAgent 177
– Erweiterte Validierung 177
5.2 WSDL Design-Ansicht 180
5.2.1 Hauptfenster 182
5.2.2 Eingabehilfen 185

6 Authentic-Ansicht 188

6.1 Benutzeroberfläche 190
6.1.1 Übersicht über die Benutzeroberfläche 190
6.1.2 Authentic-Ansicht Symbolleistenschaltflächen 191
6.1.3 Authentic-Ansicht Hauptfenster 193
6.1.4 Authentic-Ansicht Eingabehilfen 195
6.1.5 Authentic-Ansicht Kontextmenüs 199
6.2 Bearbeitung in der Authentic-Ansicht 202
6.2.1 Tabellen in der Authentic-Ansicht 202
– SPS-Tabellen 202
– XML-Tabellen 203
– Symbole zum Bearbeiten von XML-Tabellen 207
6.2.2 Bearbeiten einer DB 209

– Navigieren in einer DB-Tabelle ... 209
– DB-Abfragen ... 210
– Ändern einer DB-Tabelle ... 214
6.2.3 Arbeiten mit Datumsangaben ... 215
– Datumswahl ... 216
– Texteintrag ... 216
6.2.4 Definieren von Entities ... 217
6.2.5 Grafiken in der Authentic-Ansicht ... 219
6.2.6 Verwendung von Tasten in der Authentic-Ansicht ... 219

7 Browser-Ansicht 220

8 Archivansicht 221

8.1 Benutzeroberfläche ... 222
8.2 Operationen ... 224

9 XMLSpy in MS Visual Studio .NET 226

9.1 Unterschiede zwischen der .NET und der Standalone-Version ... 227
9.2 Visual Studio .NET und XSLT Debugger ... 231
9.3 Visual Studio .NET und SOAP Debugger ... 234

10 XMLSpy in Eclipse 236

10.1 Starten von Eclipse und Verwendung des XMLSpy Plug-Ins ... 237
10.2 Erstellen von XML-Dateien in Eclipse ... 239
10.3 Importieren von XML-Dateien in Eclipse-Ordner ... 241
10.4 Unterschiede zwischen Eclipse und den Standalone-Versionen ... 244
10.5 Eclipse-Plattform und XSLT Debugger ... 245
10.6 Eclipse Plattform und SOAP Debugger ... 247
10.7 Eclipse-Ansichten und -Perspektiven ... 250

11 XSLT und XQuery Debugger 252

11.1 Funktionsweise und Benutzeroberfläche ... 254
11.2 Befehle und Symbolleistenschaltflächen ... 256
11.3 Einstellungen ... 260
11.4 Starten einer Debugger-Sitzung ... 262
11.5 Informationsfenster ... 264
11.5.1 Kontext ... 265
11.5.2 Variablen ... 265

11.5.3 XPath-Watch ... 266
11.5.4 Call Stack ... 266
11.5.5 Messages ... 267
11.5.6 Templates ... 267
11.5.7 Info ... 268
11.5.8 Trace ... 268
11.5.9 Anordnen der Informationsfenster ... 268
11.6 Breakpoints ... 271
11.7 Tracepoints ... 274

12 XSLT/XQuery Profiler 279

12.1 Analysieren von XSLT-Transformationen ... 284
12.2 Analysieren einer XQuery-Ausführung ... 288

13 SOAP Debugger 292

13.1 SOAP-Kommunikationsvorgang ... 293
13.2 Verwendung des SOAP Debuggers ... 294
13.3 Setzen von Breakpoints im SOAP Debugger ... 300

14 Code Generator 303

14.1 Introduction to code generator ... 304
14.2 What's new in version 2007r3 ... 306
14.3 Generating source code from a schema ... 307
14.4 Using the generated code library ... 309
14.4.1 Example schema ... 310
14.4.2 Using the generated Java library ... 311
14.4.3 Using the generated C++ library ... 315
14.4.4 Using the generated C# library ... 319
14.4.5 Using generated code compatible to old versions ... 323
– Creating XML files (XMLSpy 2006) ... 324
– Creating XML files (XMLSpy 2005) ... 327
– Opening and parsing existing XML files (XMLSpy 2006) ... 329
– Opening and parsing existing XML files (XMLSpy 2005) ... 331
14.5 Code generation tips ... 334
14.6 Code generator options ... 336
14.7 The way to SPL (Spy Programming Language) ... 337
14.7.1 Basic SPL structure ... 337
14.7.2 Declarations ... 338
14.7.3 Variables ... 339
14.7.4 Predefined variables ... 340
14.7.5 Creating output files ... 341

14.7.6 Operators ... 342
14.7.7 Conditions ... 342
14.7.8 foreach ... 343
14.7.9 Subroutines ... 344
– Subroutine declaration ... 344
– Subroutine invocation ... 345
– Subroutine example ... 346
14.7.10 Built in Types ... 347
– Library ... 347
– Namespace ... 347
– Type ... 348
– Member ... 348
– NativeBinding ... 349
– Facets ... 349
– Old object model (up to v2007) ... 350
Namespace ... 350
Class ... 350
Member ... 351
Facet ... 353
Enumeration ... 353
Pattern ... 353
14.8 Error Codes ... 354

15 Benutzerreferenz 355

15.1 Menü "Datei" ... 356
15.1.1 Neu... ... 356
15.1.2 Öffnen ... 359
15.1.3 URL öffnen ... 359
15.1.4 Neu laden ... 361
15.1.5 Kodierung ... 362
15.1.6 Schließen ... 362
15.1.7 Alle schließen ... 362
15.1.8 Speichern ... 363
15.1.9 Speichern unter... ... 363
15.1.10 Speichern unter URL ... 363
15.1.11 Alles speichern ... 364
15.1.12 Als Mail senden... ... 365
15.1.13 Drucken ... 366
15.1.14 Druckvorschau ... 367
15.1.15 Druckereinrichtung... ... 367
15.1.16 Zuletzt verwendete Dateien ... 368
15.1.17 Beenden ... 368
15.2 Menü "Bearbeiten" ... 369
15.2.1 Rückgängig ... 369

15.2.2 Wiederherstellen ... 369
15.2.3 Ausschneiden ... 369
15.2.4 Kopieren ... 369
15.2.5 Einfügen ... 370
15.2.6 Löschen ... 370
15.2.7 Als XML-Text kopieren ... 370
15.2.8 Als strukturierten Text kopieren ... 370
15.2.9 Dateipfad einfügen ... 372
15.2.10 XPath kopieren ... 373
15.2.11 Pretty-Print ... 373
15.2.12 Alles markieren ... 373
15.2.13 Suchen ... 373
15.2.14 Weitersuchen ... 377
15.2.15 Ersetzen ... 377
15.2.16 In Dateien suchen ... 378
15.2.17 Lesezeichen einfügen/löschen ... 380
15.2.18 Alle Lesezeichen löschen ... 380
15.2.19 Zum nächsten Lesezeichen ... 380
15.2.20 Zum vorhergehenden Lesezeichen ... 380
15.3 Menü "Projekt" ... 382
15.3.1 Neu ... 383
15.3.2 Öffnen ... 383
15.3.3 Neu laden ... 383
15.3.4 Schließen ... 383
15.3.5 Speichern ... 383
15.3.6 Versionskontrolle ... 383
– Öffnen ... 385
– Versionskontrolle aktivieren ... 387
– Aktuellste Version holen ... 388
– Auschecken ... 388
– Einchecken ... 389
– Auschecken rückgängig ... 390
– Zu Versionskontrolle hinzufügen ... 391
– Von Versionskontrolle ausgliedern ... 392
– Verlauf anzeigen ... 393
– Unterschiede anzeigen ... 394
– Eigenschaften ... 395
– Status aktualisieren ... 396
– Native Oberfläche ausführen ... 396
15.3.7 Dateien hinzufügen ... 396
15.3.8 URL hinzufügen ... 396
15.3.9 Aktive Datei hinzufügen ... 397
15.3.10 Aktive und verwandte Dateien hinzufügen ... 397
15.3.11 Projektordner in Projekt einfügen ... 397
15.3.12 Externes Verzeichnis zum Projekt hinzufügen ... 397

15.3.13 Externes Webverzeichnis zum Projekt hinzufügen ... 400
15.3.14 Projekteigenschaften ... 402
15.3.15 Zuletzt verwendete Projekte ... 404
15.4 Menü "XML" ... 405
15.4.1 Einfügen ... 405
– Attribut einfügen ... 405
– Element einfügen ... 405
– Text einfügen ... 405
– CDATA einfügen ... 406
– Comment einfügen ... 406
– XML einfügen ... 406
– Processing Instruction einfügen ... 406
– DOCTYPE einfügen ... 406
– ExternalID einfügen ... 407
– ELEMENT einfügen ... 408
– ATTLIST einfügen ... 408
– ENTITY einfügen ... 408
– NOTATION einfügen ... 408
15.4.2 Anhängen ... 408
– Attribut anhängen ... 409
– Element anhängen ... 409
– Text anhängen ... 409
– CDATA anhängen ... 410
– Comment anhängen ... 410
– XML anhängen ... 410
– Processing Instruction anhängen ... 410
– DOCTYPE anhängen ... 410
– ExternalID anhängen ... 411
– ELEMENT anhängen ... 412
– ATTLIST anhängen ... 412
– ENTITY anhängen ... 412
– NOTATION anhängen ... 412
15.4.3 Child hinzufügen ... 412
– Child-Attribut hinzufügen ... 413
– Child-Element hinzufügen ... 413
– Child-Text hinzufügen ... 413
– Child CDATA hinzufügen ... 413
– Child Comment hinzufügen ... 414
– Child XML hinzufügen ... 414
– Child Processsing Instruction hinzufügen ... 414
– Child DOCTYPE hinzufügen ... 414
– Child ExternalID hinzufügen ... 414
– Child ELEMENT hinzufügen ... 415
– Child ATTLIST hinzufügen ... 415
– Child ENTITY hinzufügen ... 415

– Child NOTATION hinzufügen ... 415
15.4.4 Konvertieren in ... 416
– In Attribut konvertieren ... 416
– In Element konvertieren ... 416
– In Text konvertieren ... 416
– In CDATA konvertieren ... 416
– In Comment konvertieren ... 416
– In XML konvertieren ... 417
– In Processing Instruction konvertieren ... 417
– In DOCTYPE konvertieren ... 417
– In ExternalID konvertieren ... 417
– In ELEMENT konvertieren ... 417
– In ATTLIST konvertieren ... 417
– In ENTITY konvertieren ... 417
– In NOTATION konvertieren ... 417
15.4.5 Tabelle ... 418
– Als Tabelle anzeigen ... 418
– Zeile einfügen ... 419
– Zeile anhängen ... 419
– Aufsteigend sortieren ... 419
– Absteigend sortieren ... 419
15.4.6 Höher stufen ... 420
15.4.7 Tiefer stufen ... 420
15.4.8 In Element einbinden ... 420
15.4.9 XPath auswerten ... 421
15.4.10 Wohlgeformtheit prüfen ... 421
15.4.11 Validieren ... 422
15.4.12 Validieren von WSDL-Dateien ... 426
15.4.13 Eingabehilfen aktualisieren ... 427
15.4.14 Namespace Präfix ... 427
15.5 Menü "DTD/Schema" ... 428
15.5.1 DTD zuweisen ... 428
15.5.2 Schema zuweisen ... 428
15.5.3 Weitere DTD einfügen ... 428
15.5.4 Gehe zu DTD ... 429
15.5.5 Gehe zu Schema ... 429
15.5.6 Gehe zu Definition ... 429
15.5.7 DTD/Schema generieren ... 429
15.5.8 DTD/Schema konvertieren ... 431
15.5.9 Im UML konvertieren ... 434
15.5.10 In MapForce auf andere(s) DTD/Schema oder DB mappen ... 435
15.5.11 HTML/PDF-Ausgabe in StyleVision erstellen ... 435
15.5.12 XML Beispieldatei erzeugen ... 435
15.5.13 Programmcode generieren ... 437
15.5.14 Cache-Speicher leeren ... 437

15.6 Menü "Schema-Design" 438
15.6.1 Schema-Einstellungen 438
15.6.2 Diagramm speichern 439
15.6.3 Schema-Dokumentation generieren 439
15.6.4 Ansicht konfigurieren 441
15.6.5 Vergrößern/Verkleinern 444
15.6.6 Globale Elemente anzeigen 445
15.6.7 Diagramm anzeigen 445
15.6.8 Oracle Schema Extensions aktivieren 445
15.6.9 Oracle Schema-Einstellungen 446
15.6.10 Microsoft SQL Server Schema Extensions aktivieren 446
15.6.11 Named Schema Relationships 447
15.6.12 Unnamed Element Relationships 448
15.6.13 Tamino Schema Extensions aktivieren 448
15.6.14 Tamino Schema-Eigenschaften 449
15.6.15 Verbindung zu SchemaAgent Server herstellen 449
15.6.16 Verbindung mit SchemaAgent Server trennen 450
15.6.17 In SchemaAgent anzeigen 450
15.6.18 Erweiterte Validierung 451
15.7 Menü "XSL/XQuery" 452
15.7.1 XSL-Transformation 453
15.7.2 XSL:FO-Transformation 454
15.7.3 XSL-Parameter/XQuery-Variablen 455
15.7.4 XSLT 2 / XQuery-Profilerstellung aktivieren 459
15.7.5 XQuery-Ausführung 459
15.7.6 XSL zuweisen 459
15.7.7 XSL:FO zuweisen 460
15.7.8 XML-Beispieldatei zuweisen 460
15.7.9 Gehe zu XSL 460
15.7.10 Debugger starten 461
15.7.11 Debugger anhalten 461
15.7.12 Debugger neu starten 461
15.7.13 Debugger-Sitzung beenden 461
15.7.14 Einsteigen 461
15.7.15 Aussteigen 462
15.7.16 Überspringen 462
15.7.17 Aktuellen Execution Node einblenden 462
15.7.18 Breakpoint einfügen/löschen 462
15.7.19 Tracepoint einfügen/löschen 462
15.7.20 Breakpoint aktivieren/deaktivieren 463
15.7.21 Tracepoint aktivieren/deaktivieren 463
15.7.22 Breakpoints/Tracepoints 463
15.7.23 Debug-Fenster 464
15.7.24 XSLT/XQuery-Einstellungen 464
15.8 Menü "Authentic" 466

15.8.1 Neues Dokument ... 466
15.8.2 Datenbankdaten bearbeiten ... 467
15.8.3 StyleVision Stylesheet zuweisen ... 467
15.8.4 StyleVision Stylesheet bearbeiten ... 468
15.8.5 XML Entities definieren ... 468
15.8.6 Markupsymbole ausblenden ... 470
15.8.7 Kleine Markup-Symbole einblenden ... 470
15.8.8 Große Markup-Symbole einblenden ... 470
15.8.9 Gemischte Markup-Symbole einblenden ... 470
15.8.10 Zeile anhängen ... 470
15.8.11 Zeile einfügen ... 470
15.8.12 Zeile kopieren ... 471
15.8.13 Zeile nach oben ... 471
15.8.14 Zeile nach unten ... 471
15.8.15 Zeile löschen ... 471
15.9 Menü "DB" ... 472
15.9.1 Herstellen einer Verbindung zu einer Datenquelle ... 472
– Verbindungsassistent ... 473
– Bestehende Verbindungen ... 475
– ADO-Verbindungen ... 476
– ODBC-Verbindungen ... 480
15.9.2 Datenbank abfragen ... 482
– Datenquellen ... 483
– Browser-Fenster: Anzeigen der DB-Objekte ... 485
– Abfragefenster: Beschreibung und Funktionen ... 489
– Abfragefenster: Arbeiten mit Abfragen ... 492
– Ergebnisse und Meldungen ... 492
15.9.3 IBM DB2 ... 495
– XML-Schemas verwalten ... 495
– XML-Schema zuweisen ... 499
15.9.4 Tamino ... 500
– Installation von Tamino ... 501
– Installation von WebDAV ... 501
– Tamino Schema-Extensions ... 501
– Erstellen, Speichern und Auflisten von Tamino Schemas ... 502
Legacy Schemas in Tamino hinzufügen ... 508
Einschränkungen für Tamino Schema Definitionen ... 509
– Bearbeiten von Tamino Schemas ... 510
– Erstellen von XML-Dateien, die auf Tamino Schemas basieren ... 510
– Abfrage von Daten in Tamino ... 512
15.9.5 Oracle XML DB ... 516
– Suchen ... 516
– Schemas auflisten ... 518
– Schema hinzufügen ... 519
– Oracle XML-Dokumente durchsuchen ... 520

– Eigenschaften ... 521
– XML Dokumentinstanzen hinzufügen ... 522
15.9.6 Datenbank abfragen ... 522
15.10 Menü "Konvertieren" ... 523
15.10.1 Textdatei importieren ... 523
15.10.2 Datenbankdaten importieren ... 525
15.10.3 MS Word Dokument Importieren ... 529
15.10.4 XML-Schema von DB-Struktur erzeugen ... 529
15.10.5 DB-Import auf Basis von XML-Schema ... 533
15.10.6 DB Struktur von XML-Schema erzeugen ... 534
15.10.7 Export in Textdateien ... 536
15.10.8 Export in Datenbank ... 539
15.11 Menü "Ansicht" ... 542
15.11.1 Text-Ansicht ... 542
15.11.2 Enhanced Grid Ansicht ... 542
15.11.3 Schema/ WSDL Design-Ansicht ... 543
15.11.4 Authentic-Ansicht ... 543
15.11.5 Browser-Ansicht ... 543
15.11.6 Erweitern ... 543
15.11.7 Reduzieren ... 544
15.11.8 Alles erweitern ... 544
15.11.9 Unselektierte reduzieren ... 544
15.11.10 Optimale Breite ... 544
15.11.11 Zeilenumbruch ... 544
15.11.12 Gehe zu Zeile/Zeichen ... 545
15.11.13 Gehe zu Datei ... 545
15.11.14 Zeilennummerierungsleiste ... 545
15.11.15 Lesezeichenleiste ... 546
15.11.16 Klappleiste ... 546
15.11.17 Einrücklinien ... 546
15.12 Menü "Browser" ... 547
15.12.1 Zurück ... 547
15.12.2 Vorwärts ... 547
15.12.3 Abbrechen ... 547
15.12.4 Aktualisieren ... 547
15.12.5 Schriftgröße ... 547
15.12.6 Neues Fenster ... 547
15.13 Menü "WSDL" ... 548
15.13.1 Messages ... 548
15.13.2 Operations ... 548
15.13.3 PortType ... 548
15.13.4 Binding ... 549
15.13.5 Service ... 549
15.13.6 Types ... 549
15.13.7 Diagramm speichern, Dokumentation generieren ... 550

15.13.8 WSDL-Dokument neu parsen ... 551
15.14 Menü "Soap" ... 552
15.14.1 SOAP Request erzeugen ... 552
15.14.2 SOAP Request an Server senden ... 553
15.14.3 SOAP Request-Parameter ändern ... 555
15.14.4 Soap Debugger-Sitzung ... 556
15.14.5 Starten ... 557
15.14.6 Einzelschritt ... 557
15.14.7 Beim nächsten Request anhalten ... 557
15.14.8 Bei nächstem Response anhalten ... 557
15.14.9 Proxyserver anhalten ... 557
15.14.10 Soap Debugger Einstellungen ... 557
15.15 Menü "Extras" ... 559
15.15.1 Rechtschreibung ... 559
15.15.2 Rechtschreibung - Optionen ... 561
15.15.3 Skripting-Umgebung ... 565
15.15.4 Makros ... 565
15.15.5 Projekt ... 566
– Skripts zuweisen ... 566
– Projekt-Skripts entkoppeln ... 566
– Projekt-Skripts aktivieren ... 566
15.15.6 Vergleiche ... 566
– Geöffnete Datei vergleichen mit ... 566
– Verzeichnisse vergleichen ... 569
– Vergleichsoptionen ... 571
15.15.7 Anpassen ... 574
– Befehle ... 574
– Symbolleisten ... 575
– Tastatur ... 577
– Menü ... 581
– Makros ... 583
– Plug-Ins ... 584
– Optionen ... 585
– Kontextmenü 'Anpassen' ... 585
15.15.8 Optionen ... 587
– Datei ... 587
– Dateiarten ... 589
– Bearbeiten ... 591
– Ansicht ... 592
– Grid-Schriftarten ... 593
– Schemaschriftarten ... 593
– WSDL-Schriftarten ... 594
– Textschriftarten ... 595
– Farben ... 596
– Kodierung ... 598

– XSL ... 599
– Skripting-Umgebung ... 600
15.16 Menü "Fenster" ... 602
15.16.1 Überlappend ... 602
15.16.2 Horizontal anordnen ... 602
15.16.3 Vertikal anordnen ... 602
15.16.4 Projekt-Fenster ... 602
15.16.5 Info-Fenster ... 602
15.16.6 Eingabehilfen ... 602
15.16.7 Ausgabefenster ... 602
15.16.8 Projekt- und Eingabehilfen ... 603
15.16.9 Alle ein/aus ... 603
15.16.10 Liste aller offenen Fenster ... 603
15.17 Menü "Hilfe" ... 604
15.17.1 Inhaltsverzeichnis ... 604
15.17.2 Index ... 604
15.17.3 Suche ... 604
15.17.4 Tastaturzuordnungen ... 605
15.17.5 Aktivierung, Bestellformular, Registrieren, Updates ... 606
15.17.6 Support Center, Fragen und Antworten, Komponenten downloaden ... 607
15.17.7 Internet ... 607
15.17.8 Über... ... 607

Referenz für Programmierer 610

1 Release Notes 611

2 Scripting 613

2.1 Overview ... 614
2.1.1 Scripting projects ... 614
2.1.2 Running Forms, Event Handlers, and Macros ... 615
2.1.3 Scripting Options ... 617
2.2 The Scripting Environment GUI ... 618
2.2.1 Project Window ... 618
2.2.2 Main Window ... 620
2.2.3 Form Object Bar ... 621
2.2.4 The Form Editor ... 622
2.3 Using the Scripting Environment ... 627

2.3.1 Creating an Event Handler 627
2.3.2 Creating a Form 629
2.3.3 Writing a Macro 634
2.3.4 Calling macros 636
2.3.5 Programming points to note 638
2.4 Menus 639
2.4.1 File 639
2.4.2 Edit 639
2.4.3 View 639
2.4.4 Project 640
2.4.5 Layout 641
2.4.6 Draw 644
2.4.7 Window 645

3 XMLSpy Plugins 646

3.1 Registration of IDE PlugIns 647
3.2 ActiveX Controls 648
3.3 Configuration XML 649
3.4 ATL sample files 652
3.4.1 Interface description (IDL) 652
3.4.2 Class definition 654
3.4.3 Implementation 654
3.5 IXMLSpyPlugIn 657
3.5.1 OnCommand 657
3.5.2 OnUpdateCommand 658
3.5.3 OnEvent 658
3.5.4 GetUIModifications 660
3.5.5 GetDescription 661

4 The XMLSpy API 662

4.1 Overview 663
4.1.1 Object model 663
4.1.2 Simple document access 664
4.1.3 Error handling 665
4.1.4 Events 667
4.1.5 Import and export of data 670
4.1.6 Using XMLData to modify document structure 672
4.1.7 The DOM and XMLData 676
4.1.8 Obsolete Authentic Row operations 678
4.1.9 Obsolete Authentic Editing operations 679
4.2 Usage Examples 681
4.2.1 JScript: Bubble Sort Dynamic Tables 681

4.2.2 VBScript: Using object-level events ... 682
4.3 Interfaces ... 684
4.3.1 Application ... 684
– Events ... 685
OnBeforeOpenDocument ... 685
OnBeforeOpenProject ... 686
OnDocumentOpened ... 686
OnProjectOpened ... 687
– ActiveDocument ... 687
– AddMacroMenuItem ... 688
– Application ... 688
– ClearMacroMenu ... 688
– CurrentProject ... 688
– Dialogs ... 689
– Documents ... 689
– FindInFiles ... 689
– GetDatabaseImportElementList ... 689
– GetDatabaseSettings ... 690
– GetDatabaseTables ... 690
– GetExportSettings ... 691
– GetTextImportElementList ... 691
– GetTextImportExportSettings ... 692
– ImportFromDatabase ... 692
– ImportFromSchema ... 693
– ImportFromText ... 694
– ImportFromWord ... 695
– NewProject ... 695
– OpenProject ... 695
– Parent ... 696
– Quit ... 696
– ReloadSettings ... 696
– RunMacro ... 697
– ScriptingEnvironment ... 697
– ShowApplication ... 697
– ShowFindInFiles ... 698
– ShowForm ... 698
– URLDelete ... 698
– URLMakeDirectory ... 699
– Visible ... 699
– WarningNumber ... 699
– WarningText ... 699
4.3.2 AuthenticDataTransfer ... 700
– dropEffect ... 700
– getData ... 701
– ownDrag ... 701

– type ... 701
4.3.3 AuthenticRange ... 701
– AppendRow ... 703
– Application ... 703
– CanPerformAction ... 704
– CanPerformActionWith ... 704
– Clone ... 705
– CollapsToBegin ... 705
– CollapsToEnd ... 705
– Copy ... 705
– Cut ... 706
– Delete ... 706
– DeleteRow ... 706
– DuplicateRow ... 707
– ExpandTo ... 707
– FirstTextPosition ... 707
– FirstXMLData ... 708
– FirstXMLDataOffset ... 709
– GetElementAttributeNames ... 710
– GetElementAttributeValue ... 710
– GetElementHierarchy ... 711
– GetEntityNames ... 711
– Goto ... 711
– GotoNext ... 712
– GotoNextCursorPosition ... 712
– GotoPrevious ... 713
– GotoPreviousCursorPosition ... 713
– HasElementAttribute ... 714
– InsertEntity ... 714
– InsertRow ... 715
– IsCopyEnabled ... 715
– IsCutEnabled ... 715
– IsDeleteEnabled ... 716
– IsEmpty ... 716
– IsEqual ... 716
– IsFirstRow ... 716
– IsInDynamicTable ... 717
– IsLastRow ... 717
– IsPasteEnabled ... 717
– IsTextStateApplied ... 717
– LastTextPosition ... 718
– LastXMLData ... 718
– LastXMLDataOffset ... 719
– MoveBegin ... 720
– MoveEnd ... 720

– MoveRowDown ... 721
– MoveRowUp ... 721
– Parent ... 721
– Paste ... 722
– PerformAction ... 722
– Select ... 723
– SelectNext ... 723
– SelectPrevious ... 724
– SetElementAttributeValue ... 724
– SetFromRange ... 725
– Text ... 726
4.3.4 AuthenticView ... 726
– Events ... 727
OnBeforeCopy ... 727
OnBeforeCut ... 728
OnBeforeDelete ... 728
OnBeforeDrop ... 728
OnBeforePaste ... 730
OnDragOver ... 730
OnKeyboardEvent ... 731
OnMouseEvent ... 732
OnSelectionChanged ... 734
– Application ... 735
– AsXMLString ... 735
– DocumentBegin ... 735
– DocumentEnd ... 735
– Event ... 736
– Goto ... 736
– IsRedoEnabled ... 737
– IsUndoEnabled ... 737
– MarkupVisibility ... 737
– Parent ... 737
– Print ... 738
– Redo ... 738
– Selection ... 738
– Undo ... 739
– UpdateXMLInstanceEntities ... 739
– WholeDocument ... 739
– XMLDataRoot ... 740
4.3.5 CodeGeneratorDlg ... 740
– Application ... 741
– CompatibilityMode ... 741
– CPPSettings_DOMType ... 741
– CPPSettings_GenerateVC6ProjectFile ... 742
– CPPSettings_GenerateVSProjectFile ... 742

– CPPSettings_LibraryType ... 742
– CPPSettings_UseMFC ... 742
– CSharpSettings_ProjectType ... 743
– OutputPath ... 743
– OutputPathDialogAction ... 743
– OutputResultDialogAction ... 743
– Parent ... 744
– ProgrammingLanguage ... 744
– PropertySheetDialogAction ... 744
– TemplateFileName ... 745
4.3.6 DatabaseConnection ... 745
– ADOConnection ... 745
– AsAttributes ... 746
– CommentIncluded ... 746
– CreateMissingTables ... 746
– CreateNew ... 747
– DatabaseKind ... 747
– ExcludeKeys ... 747
– File ... 747
– IncludeEmptyElements ... 748
– NullReplacement ... 748
– NumberDateTimeFormat ... 748
– ODBCConnection ... 748
– SQLSelect ... 749
– TextFieldLen ... 749
4.3.7 Dialogs ... 749
– Application ... 750
– CodeGeneratorDlg ... 750
– FileSelectionDlg ... 750
– Parent ... 750
– SchemaDocumentationDlg ... 751
– GenerateSampleXMLDlg ... 751
– DTDSchemaGeneratorDlg ... 751
– FindInFilesDlg ... 751
– WSDLDocumentationDlg ... 752
4.3.8 Document ... 752
– Events ... 754
OnBeforeSaveDocument ... 754
OnBeforeCloseDocument ... 754
OnBeforeValidate ... 755
OnCloseDocument ... 755
OnViewActivation ... 756
– Application ... 756
– AssignDTD ... 757
– AssignSchema ... 757

– AssignXSL ... 757
– AssignXSLFO ... 757
– AuthenticView ... 758
– Close ... 758
– ConvertDTDOrSchema ... 759
– ConvertDTDOrSchemaEx ... 759
– CreateChild ... 760
– CreateSchemaDiagram ... 760
– CurrentViewMode ... 760
– DataRoot ... 761
– DocEditView ... 761
– Encoding ... 761
– EndChanges ... 762
– ExecuteXQuery ... 762
– ExportToDatabase ... 762
– ExportToText ... 763
– FullName ... 764
– GenerateDTDOrSchema ... 764
– GenerateDTDOrSchemaEx ... 765
– GenerateProgramCode ... 765
– GenerateSampleXML ... 765
– GenerateSchemaDocumentation ... 766
– GenerateWSDLDocumentation ... 766
– GetExportElementList ... 766
– GetPathName (obsolete) ... 767
– GridView ... 767
– IsModified ... 767
– IsValid ... 768
– IsWellFormed ... 769
– Name ... 769
– Parent ... 770
– Path ... 770
– RootElement ... 770
– Save ... 770
– SaveAs ... 771
– Saved ... 771
– SaveInString ... 771
– SaveToURL ... 772
– SetActiveDocument ... 772
– SetEncoding (obsolete) ... 772
– SetExternalIsValid ... 774
– SetPathName (obsolete) ... 774
– StartChanges ... 774
– Suggestions ... 775
– SwitchViewMode ... 775

– TextView 775
– Title 775
– TransformXSL 776
– TransformXSLFO 776
– UpdateViews 776
– UpdateXMLData 776
4.3.9 Documents 777
– Count 777
– Item 778
– NewFile 778
– NewFileFromText 778
– OpenFile 779
– OpenURL 779
– OpenURLDialog 780
4.3.10 DTDSchemaGeneratorDlg 780
– Application 781
– AttributeTypeDefinition 781
– DTDSchemaFormat 781
– FrequentElements 781
– GlobalAttributes 781
– MaxEnumLength 782
– MergeAllEqualNamed 782
– OnlyStringEnums 782
– OutputPath 782
– OutputPathDialogAction 783
– Parent 783
– ResolveEntities 783
– TypeDetection 783
– ValueList 783
4.3.11 ElementList 784
– Count 784
– Item 784
– RemoveElement 784
4.3.12 ElementListItem 785
– ElementKind 785
– FieldCount 785
– Name 785
– RecordCount 785
4.3.13 ExportSettings 786
– CreateKeys 786
– ElementList 786
– EntitiesToText 787
– ExportAllElements 787
– ExportCompleteXML 787
– FromAttributes 787

– FromSingleSubElements ... 787
– FromTextValues ... 787
– IndependentPrimaryKey ... 788
– Namespace ... 788
– StartFromElement ... 788
– SubLevelLimit ... 788
4.3.14 FileSelectionDlg ... 788
– Application ... 789
– DialogAction ... 789
– FullName ... 789
– Parent ... 790
4.3.15 FindInFilesDlg ... 790
– AdvancedXMLSearch ... 790
– Application ... 791
– DoReplace ... 791
– FileExtension ... 791
– Find ... 791
– IncludeSubfolders ... 792
– MatchCase ... 792
– MatchWholeWord ... 792
– Parent ... 792
– RegularExpression ... 792
– Replace ... 793
– ReplaceOnDisk ... 793
– SearchInProjectFilesDoExternal ... 793
– SearchLocation ... 793
– ShowResult ... 793
– StartFolder ... 794
– XMLAttributeContents ... 794
– XMLAttributeNames ... 794
– XMLCData ... 794
– XMLComments ... 794
– XMLElementContents ... 795
– XMLElementNames ... 795
– XMLPI ... 795
– XMLRest ... 795
4.3.16 FindInFilesResult ... 795
– Application ... 796
– Count ... 796
– Document ... 796
– Item ... 796
– Parent ... 796
– Path ... 797
4.3.17 FindInFilesResultMatch ... 797
– Application ... 797

– Length 797
– Line 798
– LineText 798
– Parent 798
– Position 798
– Replaced 799
4.3.18 FindInFilesResults 799
– Application 799
– Count 799
– Item 799
– Parent 800
4.3.19 GenerateSampleXMLDlg 800
– Application 800
– Parent 801
– NonMandatoryAttributes 801
– NonMandatoryElements - obsolete 801
– TakeFirstChoice - obsolete 801
– RepeatCount 801
– FillWithSampleData - obsolete 802
– FillElementsWithSampleData 802
– FillAttributesWithSampleData 802
– ContentOfNillableElementsIsNonMandatory 802
– TryToUseNonAbstractTypes 802
– Optimization 803
– SchemaOrDTDAssignment 803
– LocalNameOfRootElement 803
– NamespaceURIOfRootElement 803
– OptionsDialogAction 803
4.3.20 GridView 804
– Events 804
OnBeforeDrag 804
OnBeforeDrop 804
OnBeforeStartEditing 805
OnEditingFinished 805
OnFocusChanged 806
– CurrentFocus 806
– Deselect 806
– IsVisible 807
– Select 807
– SetFocus 807
4.3.21 SchemaDocumentationDlg 807
– AllDetails 808
– Application 808
– DiagramFormat 809
– EmbedDiagrams 809

– IncludeAll ... 809
– IncludeAttributeGroups ... 810
– IncludeComplexTypes ... 810
– IncludeGlobalAttributes ... 810
– IncludeGlobalElements ... 810
– IncludeGroups ... 811
– IncludeIndex ... 811
– IncludeLocalAttributes ... 811
– IncludeLocalElements ... 811
– IncludeRedefines ... 812
– IncludeReferencedSchemas ... 812
– IncludeSimpleTypes ... 812
– MultipleOutputFiles ... 813
– OptionsDialogAction ... 813
– OutputFile ... 813
– OutputFileDialogAction ... 813
– OutputFormat ... 814
– Parent ... 814
– ShowAnnotations ... 814
– ShowAttributes ... 815
– ShowChildren ... 815
– ShowDiagram ... 815
– ShowEnumerations ... 815
– ShowIdentityConstraints ... 816
– ShowNamespace ... 816
– ShowPatterns ... 816
– ShowProgressBar ... 817
– ShowProperties ... 817
– ShowResult ... 817
– ShowSingleFacets ... 817
– ShowSourceCode ... 818
– ShowType ... 818
– ShowUsedBy ... 818

4.3.22 SpyProject ... 819
– CloseProject ... 819
– ProjectFile ... 819
– RootItems ... 819
– SaveProject ... 820
– SaveProjectAs ... 820

4.3.23 SpyProjectItem ... 820
– ChildItems ... 820
– FileExtensions ... 821
– ItemType ... 821
– Name ... 821
– Open ... 821

– ParentItem ... 821
– Path ... 822
– ValidateWith ... 822
– XMLForXSLTransformation ... 822
– XSLForXMLTransformation ... 822
– XSLTransformationFileExtension ... 822
– XSLTransformationFolder ... 822
4.3.24 SpyProjectItems ... 823
– AddFile ... 823
– AddFolder ... 823
– AddURL ... 823
– Count ... 824
– Item ... 824
– RemoveItem ... 824
4.3.25 TextImportExportSettings ... 824
– CommentIncluded ... 825
– DestinationFolder ... 825
– EnclosingCharacter ... 825
– Encoding ... 825
– EncodingByteOrder ... 826
– FieldDelimiter ... 826
– FileExtension ... 826
– HeaderRow ... 826
– ImportFile ... 826
– RemoveDelimiter ... 827
– RemoveNewline ... 827
4.3.26 TextView ... 827
– Events ... 827
OnBeforeShowSuggestions ... 827
OnChar ... 828
– Application ... 829
– GetRangeText ... 829
– GoToLineChar ... 829
– Length ... 829
– LineCount ... 829
– LineFromPosition ... 830
– LineLength ... 830
– MoveCaret ... 830
– Parent ... 830
– PositionFromLine ... 830
– ReplaceText ... 831
– SelectionEnd ... 831
– SelectionStart ... 831
– SelectText ... 831
– SelText ... 831

– Text ... 832
4.3.27 WSDLDocumentationDlg ... 832
– AllDetails ... 833
– Application ... 833
– DiagramFormat ... 833
– EmbedDiagrams ... 834
– IncludeAll ... 834
– IncludeBinding ... 834
– IncludeImportedWSDLFiles ... 834
– IncludeMessages ... 835
– IncludeOverview ... 835
– IncludePortType ... 835
– IncludeService ... 835
– IncludeTypes ... 836
– MultipleOutputFiles ... 836
– OptionsDialogAction ... 836
– OutputFile ... 837
– OutputFileDialogAction ... 837
– OutputFormat ... 837
– Parent ... 838
– ShowBindingDiagram ... 838
– ShowExtensibility ... 838
– ShowMessageParts ... 838
– ShowPort ... 839
– ShowPortTypeDiagram ... 839
– ShowPortTypeOperations ... 839
– ShowProgressBar ... 840
– ShowResult ... 840
– ShowServiceDiagram ... 840
– ShowSourceCode ... 840
– ShowTypesDiagram ... 841
– ShowUsedBy ... 841
4.3.28 XMLData ... 841
– AppendChild ... 842
– CountChildren ... 843
– CountChildrenKind ... 843
– EraseAllChildren ... 843
– EraseCurrentChild ... 844
– GetChild ... 844
– GetChildKind ... 845
– GetCurrentChild ... 845
– GetFirstChild ... 845
– GetNextChild ... 846
– HasChildrenKind ... 847
– HasChildren ... 847

– InsertChild ... 848
– IsSameNode ... 848
– Kind ... 848
– MayHaveChildren ... 848
– Name ... 849
– Parent ... 849
– TextValue ... 849
4.4 Interfaces (obsolete) ... 850
4.4.1 AuthenticEvent (obsolete) ... 850
– altKey (obsolete) ... 851
– altLeft (obsolete) ... 852
– button (obsolete) ... 853
– cancelBubble (obsolete) ... 854
– clientX (obsolete) ... 855
– clientY (obsolete) ... 856
– ctrlKey (obsolete) ... 857
– ctrlLeft (obsolete) ... 858
– dataTransfer (obsolete) ... 859
– fromElement (obsolete) ... 860
– keyCode (obsolete) ... 860
– propertyName (obsolete) ... 861
– repeat (obsolete) ... 861
– returnValue (obsolete) ... 861
– shiftKey (obsolete) ... 862
– shiftLeft (obsolete) ... 863
– srcElement (obsolete) ... 864
– type (obsolete) ... 865
4.4.2 AuthenticSelection (obsolete) ... 866
– End (obsolete) ... 867
– EndTextPosition (obsolete) ... 867
– Start (obsolete) ... 868
– StartTextPosition (obsolete) ... 868
4.4.3 OldAuthentictView (obsolete) ... 869
– ApplyTextState (obsolete) ... 871
– CurrentSelection (obsolete) ... 872
– EditClear (obsolete) ... 872
– EditCopy (obsolete) ... 873
– EditCut (obsolete) ... 873
– EditPaste (obsolete) ... 874
– EditRedo (obsolete) ... 874
– EditSelectAll (obsolete) ... 875
– EditUndo (obsolete) ... 875
– event (obsolete) ... 876
– GetAllowedElements (obsolete) ... 876
– GetNextVisible (obsolete) ... 879

– GetPreviousVisible (obsolete) ... 880
– IsEditClearEnabled (obsolete) ... 881
– IsEditCopyEnabled (obsolete) ... 881
– IsEditCutEnabled (obsolete) ... 882
– IsEditPasteEnabled (obsolete) ... 882
– IsEditRedoEnabled (obsolete) ... 883
– IsEditUndoEnabled (obsolete) ... 883
– IsRowAppendEnabled (obsolete) ... 884
– IsRowDeleteEnabled (obsolete) ... 884
– IsRowDuplicateEnabled (obsolete) ... 885
– IsRowInsertEnabled (obsolete) ... 885
– IsRowMoveDownEnabled (obsolete) ... 886
– IsRowMoveUpEnabled (obsolete) ... 886
– IsTextStateApplied (obsolete) ... 887
– IsTextStateEnabled (obsolete) ... 887
– LoadXML (obsolete) ... 888
– MarkUpView (obsolete) ... 888
– RowAppend (obsolete) ... 890
– RowDelete (obsolete) ... 890
– RowDuplicate (obsolete) ... 891
– RowInsert (obsolete) ... 891
– RowMoveDown (obsolete) ... 892
– RowMoveUp (obsolete) ... 892
– SaveXML (obsolete) ... 893
– SelectionMoveTabOrder (obsolete) ... 894
– SelectionSet (obsolete) ... 895
– XMLRoot (obsolete) ... 896

4.5 Enumerations ... 898
4.5.1 SPYAttributeTypeDefinition ... 898
4.5.2 SPYAuthenticActions ... 898
4.5.3 SPYAuthenticDocumentPosition ... 898
4.5.4 SPYAuthenticElementActions ... 898
4.5.5 SPYAuthenticElementKind ... 899
4.5.6 SPYAuthenticMarkupVisibility ... 899
4.5.7 SPYDatabaseKind ... 899
4.5.8 SPYDialogAction ... 899
4.5.9 SPYDOMType ... 900
4.5.10 SPYDTDSchemaFormat ... 900
4.5.11 SPYEncodingByteOrder ... 900
4.5.12 SPYExportNamespace ... 900
4.5.13 SPYFindInFilesSearchLocation ... 900
4.5.14 SPYFrequentElements ... 900
4.5.15 SPYImageKind ... 901
4.5.16 SPYKeyEvent ... 901
4.5.17 SPYLibType ... 901

4.5.18 SPYLoading....901
4.5.19 SPYMouseEvent....901
4.5.20 SPYNumberDateTimeFormat....902
4.5.21 SPYProgrammingLanguage....902
4.5.22 SPYProjectItemTypes....902
4.5.23 SPYProjectType....903
4.5.24 SPYSampleXMLGenerationOptimization....903
4.5.25 SPYSampleXMLGenerationSchemaOrDTDAssignment....903
4.5.26 SPYSchemaDefKind....903
4.5.27 SPYSchemaDocumentationFormat....904
4.5.28 SPYTextDelimiters....904
4.5.29 SPYTextEnclosing....904
4.5.30 SPYTypeDetection....904
4.5.31 SPYURLTypes....905
4.5.32 SPYViewModes....905
4.5.33 SPYVirtualKeyMask....905
4.5.34 SPYXMLDataKind....906

5 Using the XMLSpy API with Java 907

5.1 Sample source code....909
5.2 SpyApplication....911
5.3 SpyCodeGeneratorDlg....912
5.4 SpyDatabaseConnection....913
5.5 SpyDialogs....914
5.6 SpyDoc....915
5.7 SpyDocuments....917
5.8 SpyElementList....918
5.9 SpyElementListItem....919
5.10 SpyExportSettings....920
5.11 SpyFileSelectionDlg....921
5.12 SpyGridView....922
5.13 SpyProject....923
5.14 SpyProjectItem....924
5.15 SpyProjectItems....925
5.16 SpySchemaDocumentationDlg....926
5.17 SpyTextImportExportSettings....928
5.18 SpyXMLData....929
5.19 Authentic....930
5.19.1 SpyAuthenticRange....930
5.19.2 SpyAuthenticView....931
5.19.3 SpyDocEditSelection....931
5.19.4 SpyDocEditView....931
5.20 Predefined constants....933
5.20.1 SPYAuthenticActions....933

5.20.2 SPYAuthenticDocumentPosition 933
5.20.3 SPYAuthenticElementKind 933
5.20.4 SPYAuthenticMarkupVisibility 934
5.20.5 SPYDatabaseKind 934
5.20.6 SPYDialogAction 934
5.20.7 SPYDOMType 935
5.20.8 SPYDTDSchemaFormat 935
5.20.9 SPYEncodingByteOrder 935
5.20.10 SPYExportNamespace 935
5.20.11 SPYFrequentElements 935
5.20.12 SPYLibType 936
5.20.13 SPYLoading 936
5.20.14 SPYNameDateTimeFormat 936
5.20.15 SPYProgrammingLanguage 936
5.20.16 SPYProjectItemTypes 936
5.20.17 SPYProjectType 937
5.20.18 SPYSchemaDefKind 937
5.20.19 SPYSchemaDocumentationFormat 937
5.20.20 SPYTextDelimiters 938
5.20.21 SPYTextEnclosing 938
5.20.22 SPYTypeDetection 938
5.20.23 SPYURLTypes 938
5.20.24 SpyViewModes 939
5.20.25 SPYWhitespaceComparison 939
5.20.26 SPYXMLDataKind 939

6 XMLSpy Integration 941

6.1 Integration at the Application Level 942
6.1.1 Example: HTML 942
– Instantiate the Control 942
– Add Button to Open Default Document 942
– Add Buttons for Code Generation 943
– Connect to Custom Events 944
6.2 Integration at Document Level 945
6.2.1 Use XMLSpyControl 945
6.2.2 Use XMLSpyControlDocument 946
6.2.3 Use XMLSpyControlPlaceHolder 946
6.2.4 Query XMLSpy Commands 946
6.2.5 Examples 946
– C# 946
Introduction 947
Placing the XMLSpyControl 947
Adding the Document Control 947
Adding the Placeholder Control 947

Retrieving Command Information 948
Testing the Example 949
– HTML 949
Instantiate the XMLSpyControl 950
Create Editor window 950
Create Project Window 950
Create Placeholder for XMLSpy Helper Windows 950
Create a Custom Toolbar 951
Create More Buttons 951
Create Event Handler to Update Button Status 953
6.3 Command Table 954
6.3.1 File Menu 954
6.3.2 Edit Menu 955
6.3.3 Project Menu 955
6.3.4 XML menu 956
6.3.5 DTD/Schema Menu 958
6.3.6 Schema Design Menu 958
6.3.7 XSL/XQuery Menu 959
6.3.8 Authentic Menu 961
6.3.9 Convert Menu 961
6.3.10 View Menu 962
6.3.11 Browser Menu 962
6.3.12 WSDL Menu 963
6.3.13 SOAPMenu 964
6.3.14 Tools Menu 964
6.3.15 Window Menu 965
6.3.16 Help Menu 965
6.4 Accessing XMLSpy API 967
6.5 Object Reference 968
6.5.1 Command 968
– Accelerator 968
– ID 968
– IsSeparator 969
– Label 969
– StatusText 969
– SubCommands 969
– ToolTip 969
6.5.2 Commands 969
– Count 970
– Item 970
6.5.3 XMLSpyControl 970
– Properties 971
Appearance 971
Application 971
BorderStyle 971

CommandsList ... 971
CommandsStructure (deprecated) ... 972
EnableUserPrompts ... 972
IntegrationLevel ... 973
MainMenu ... 973
Toolbars ... 973
– Methods ... 973
Exec ... 973
Open ... 974
QueryStatus ... 974
– Events ... 974
OnCloseEditingWindow ... 975
OnContextChanged ... 975
OnDocumentOpened ... 975
OnFileChangedAlert ... 975
OnLicenseProblem ... 975
OnOpenedOrFocused ... 976
OnUpdateCmdUI ... 976
OnValidationWindowUpdated ... 976
6.5.4 XMLSpyControlDocument ... 976
– Properties ... 977
Appearance ... 977
BorderStyle ... 977
Document ... 978
IsModified ... 978
Path ... 978
ReadOnly ... 978
– Methods ... 978
Exec ... 979
New ... 979
Open ... 979
QueryStatus ... 979
Reload ... 980
Save ... 980
SaveAs ... 980
– Events ... 980
OnActivate ... 980
OnContextChanged ... 981
OnDocumentClosed ... 981
OnDocumentOpened ... 981
OnDocumentSaveAs ... 981
OnFileChangedAlert ... 982
OnModifiedFlagChanged ... 982
OnSetEditorTitle ... 982
6.5.5 XMLSpyControlPlaceHolder ... 982

– Properties ... 983
Label ... 983
PlaceholderWindowID ... 983
Project ... 983
– Methods ... 983
OpenProject ... 984
CloseProject ... 984
– Events ... 984
OnModifiedFlagChanged ... 984
6.5.6 Enumerations ... 984
– ICActiveXIntegrationLevel ... 984
– XMLSpyControlPlaceholderWindow ... 985

Anhänge 988

1 Prozessor-Informationen 989

1.1 XSLT 1.0-Prozessor: Implementierungsinformationen ... 990
1.2 XSLT 2.0-Prozessor: Implementierungsinformationen ... 992
1.2.1 Allgemeine Informationen ... 992
1.2.2 XSLT 2.0-Elemente und -Funktionen ... 994
1.3 XQuery 1.0-Prozessor: Implementierungsinformationen ... 995
1.4 XPath 2.0- und XQuery 1.0-Funktionen ... 998
1.4.1 Allgemeine Informationen ... 998
1.4.2 Unterstützung von Funktionen ... 999

2 Datentypen in XML-Schemas, die von DB generiert wurden 1002

2.1 MS Access ... 1003
2.2 MS SQL Server ... 1004
2.3 MySQL ... 1005
2.4 Oracle ... 1006
2.5 ODBC ... 1007
2.6 ADO ... 1008
2.7 Sybase ... 1009

3 Datentypen in DB, die von XML-Schema generiert wurden 1010

3.1 MS Access ... 1011
3.2 MS SQL Server ... 1013
3.3 MySQL ... 1015

3.4 Oracle 1017

4 Technische Daten 1019

4.1 OS- und Arbeitsspeicheranforderungen 1020
4.2 Altova XML Parser 1021
4.3 Altova XSLT- und XQuery-Prozessor 1022
4.4 Unicode-Unterstützung 1023
4.4.1 Windows 2000 und Windows XP 1023
4.4.2 Rechts nach Links-Schreibrichtung 1024
4.5 Internet-Verwendung 1025

5 Lizenzinformationen 1026

5.1 Electronic Software Distribution 1027
5.2 Lizenzüberwachung 1028
5.3 Copyright 1029
5.4 Altova Endbenutzer-Lizenzvereinbarung 1030

Index 1043

Altova XMLSpy 2007 Enterprise Edition

Willkommen bei der XMLSpy Enterprise Edition

Willkommen bei der XMLSpy Enterprise Edition

Altova XMLSpy® 2007 ist die Industriestandard XML-Entwicklungsumgebung zum Erstellen, Editieren und Debuggen von Unternehmensapplikationen, die auf XML, XML-Schema, XSLT, XQuery, SOAP, WSDL und Webservice-Technologien basieren. Mit Hilfe dieser Anwendung können J2EE-, .NET- und Datenbank-Entwickler nun schneller und effizienter arbeiten.

Copyright © 1998–2007. Altova GmbH. All rights reserved. Use of this software is governed by and subject to an Altova software license agreement. XMLSpy, MapForce, StyleVision, SemanticWorks, SchemaAgent, UModel, DatabaseSpy, DiffDog, Authentic, AltovaXML, MissionKit, and ALTOVA as well as their logos are trademarks and/or registered trademarks of Altova GmbH.

ALTOVA®

XML, XSL, XHTML, and W3C are trademarks (registered in numerous countries) of the World Wide Web Consortium; marks of the W3C are registered and held by its host institutions, MIT, INRIA, and Keio. UNICODE and the Unicode Logo are trademarks of Unicode Inc. This software contains 3rd party copyrighted software or material that is protected by copyright and subject to other terms and conditions as detailed on the Altova website at http://www.altova.com/legal_3rdparty.html

Altova XMLSpy 2007 Enterprise Edition

Benutzerhandbuch

Benutzerhandbuch

Dieses Benutzerhandbuch enthält Tutorials und Erklärungen zu den verschiedenen Features von XMLSpy, um Ihnen eine Einführung in das Programm zu geben. Außerdem finden Sie darin ein umfassendes Referenzhandbuch, in dem die Funktionen der XMLSpy IDE beschrieben sind. Das Benutzerhandbuch ist in die folgenden Abschnitte gegliedert:

- eine Einführung, die eine Übersicht über die XML-Produkte von Altova enthält.
- ein **Tutorial,** in dem beschrieben wird, wie XMLSpy für die verschiedenen Aspekte von XML verwendet wird:
  - XML-Editierung und -Validierung
  - Schema/DTD-Editierung und -Validierung
  - XSLT-Editierung und -Transformation
- ausführliche Beschreibungen der fünf Ansichten von XMLSpy: Textansicht, Enhanced Grid-Ansicht, Schema/WSDL-Ansicht, Authentic-Ansicht und Browser-Ansicht. In diesen fünf Abschnitten werden die GUI und Features der verschiedenen Ansichten sowie die verschiedenen Komponenten und Symbole der einzelnen Ansichten beschrieben und Sie erfahren, wie man mit den Funktionen dieser Ansichten arbeitet.
- eine Beschreibung der Integration von XMLSpy mit **Visual Studio .NET**.
- Eine Beschreibung der **Eclipse** Integration in XMLSpy.
- eine Beschreibung der Verwendung der **XSLT/XQuery Debugger**-Komponente, die in Ihrem XMLSpy Produkt integriert ist. Mit Hilfe des XSLT/XQuery Debuggers können Sie Ihre XSLT 1.0 und XSLT 2.0 Stylesheets und XQuery-Dokumente überprüfen und korrigieren. In einer XSLT Debugger-Sitzung können Sie die Transformation einer XML-Datei anhand eines ausgewählten Stylesheet Schritt für Schritt durchgehen. In einer XQuery Debugger-Sitzung können Sie ein XQuery-Dokument Schritt für Schritt durchgehen und eventuelle Fehler ausfindig machen.
- eine Beschreibung des **XSLT/XQuery Profilers**, der zum Analysieren der Ausführungsdauer von XSLT Stylesheets und XQuery-Dokumenten verwendet wird.
- eine Beschreibung des **SOAP Debuggers** von XMLSpy.
- eine Beschreibung der **Codegenerierungsfunktionen** von XMLSpy.
- eine **Benutzerreferenz**, die eine Beschreibung aller Menübefehle von XMLSpy enthält.

© 2007 Altova GmbH

1 Einführung

Diese Einführung enthält einen kurzen Überblick über:

- die verschiedenen XML-Produkte von Altova und wie sie einander ergänzen
- die wichtigsten Funktionen von XMLSpy
- die Benutzeroberfläche

Dieser Abschnitt soll Ihnen einen allgemeinen Überblick über XMLSpy verschaffen und Sie mit den Funktionen des Produkts und der Benutzeroberfläche vertraut machen.

1.1 Die XML-Produkte von Altova

Die XLM-Produkte von Altova sind einfach zu bedienen und decken alle XML-Anforderungen ab. Sie ergänzen einander und bieten Ihnen eine umfassende XML-Entwicklungsumgebung. Derzeit umfasst die Altova-Produktpalette die folgenden Produkte.

Altova XMLSpy® 2007 ist eine umfassende IDE zur Entwicklung von XML-Projekten und steht in drei Versionen zur Verfügung. Den größten Funktionsumfang bietet die Enterprise Edition mit Funktionen zum Erzeugen und Bearbeiten von DTDs, XML-Schemas, XML-Dateien und XSLT Stylesheets. Es stehen Ihnen leistungsstarke Editierfeatures, mehrere Dokumentenansichten - einschließlich der Altova Authentic-Ansicht - Funktionen zum Validieren und für XSLT-Transformationen mittels eines integrierten Prozessors zur Verfügung. Des Weiteren bietet das Programm Import- und Exportfunktionen für Textdateien und Datenbanken, Funktionen zur Verwaltung von XML-Dokumenten in Projekten, einen XSLT- und XQuery Debugger, einen WSDL Editor sowie Funktionen zur Codegenerierung. Die Professional Edition und die Home Edition enthalten einen anders konfigurierten Funktionsumfang.

Altova XML 2007 ist eine kostenlose Applikation, die den Altova XML-Validator sowie einen XLST 1.0-, XSLT 2.0- und XQuery 1.0-Prozessor enthält. Das Tool kann über die Befehlszeile, über eine COM-Schnittstelle, in Java-Programmen und in .NET-Applikationen zur Validierung von XML-Dokumenten, zum Transformieren von XML-Dokumenten unter Verwendung von XSLT 1.0 und 2.0 Stylesheets sowie zum Ausführen von XQuery-Dokumenten verwendet werden.

Altova StyleVision® 2007 ist eine neue Lösung zum Erstellen komplexer XSLT Stylesheets mit Hilfe einer intuitiven Drag-and-Drop-Oberfläche. Mit StyleVision lassen sich auch StyleVision Power Stylesheets erstellen, über die Sie die Anzeige und Dateneingabe in XML-Dokumente in der Authentic-Ansicht steuern können. StyleVision ist in Form einer Enterprise und einer Professional Edition erhältlich.

Altova Authentic® 2007 (Desktop und Browser Edition) Diese beiden Versionen sind einem Textverarbeitungsprogramm ähnliche Editierprogramme, die die Eingabe von Daten in XML-Dokumente mittels Formularen unterstützen. Sie können damit Komponenten wie Grafiken und Tabellen einfügen und Dateien in Echtzeit gegen ein Schema validieren.

Altova MapForce™ 2007 dient zum Mappen von einem Schema auf ein anderes und zum Konvertieren von schemabasierten XML-Dateien in XML-Dateien, die auf einem anderen Schema basieren.

Altova Website
Werfen Sie bitte auch gelegentlich einen Blick auf die Altova Website, www.altova.com, wo Sie sich über Produktneuheiten informieren können sowie Updates, zusätzliche Dokumentation und Support erhalten.

Beachten Sie bitte, dass die Benutzerhandbücher für alle Altova-Produkte in den folgenden Formaten zur Verfügung stehen:

- Online-Handbücher: stehen auf der Support-Seite der Altova Website zur Verfügung
- PDFs zum Ausdrucken, die Sie von der Altova Website herunterladen und ausdrucken können
- Druckausgabe der Handbücher, die Sie über einen Link auf der Altova Website bestellen und erwerben können.

Die Dokumentation auf der Website wird regelmäßig aktualisiert und entspricht dem Stand der aktuellen Version.

Support und Feedback

© 2007 Altova GmbH

Wenn Sie weitere Informationen benötigen oder Fragen zu unseren Produkten haben, besuchen Sie unser Support Center auf der Altova Website. Hier finden Sie:

- Links zu unseren FAQ-Seiten
- Diskussionsforen zu Altova-Produkten und allgemeinen XML-Themen
- Online Support-Formulare, die an unser Support-Team gesendet werden.

Wir freuen uns auch über Feedback zu unserer Dokumentation. Kontaktieren Sie uns unter http://www.altova.com/de/support_center.html.

1.2 Die wichtigsten Funktionen von XMLSpy

XMLSpy ist eine IDE (Integrated Development Environment) zur Entwicklung von XML-Projekten. Mit XMLSpy können Sie unter anderem die verschiedensten XML- und Textdokumente editieren und verarbeiten; Daten in XML-Dokumente importieren und aus diesen exportieren (sowie Daten in Datenbanken importieren bzw. aus diesen exportieren); Konvertierungen zwischen bestimmten Arten von XML-Dokumenten und anderen Dokumentarten vornehmen; unterschiedliche Arten von XML-Dokumenten in Projekten verknüpfen; Dokumente mittels des integrierten XSLT 1.0-, XSLT 2.0- und XQuery 1.0-Prozessors verarbeiten, und sogar Code anhand von XML-Dokumenten generieren.

XMLSpy bietet außerdem eine grafische Bearbeitungsansicht für XML-Dokumente, nämlich die beliebte Altova Authentic-Ansicht, über die Benutzer Daten ähnlich wie bei einem Textverarbeitungsprogramm in ein XML-Dokument eingeben können. Die Authentic-Ansicht eignet sich vor allem für folgende Zwecke:

- wenn Mitarbeiter ohne XML-Kenntnisse Daten in XML-Dokumente eingeben müssen
- wenn mehrere Benutzer Daten in ein einziges Dokument eingeben wollen/auf dieses zugreifen wollen, das über einen Server oder eine gemeinsam benutzte Ressource zugänglich ist.

Dieser Abschnitt enthält einen kurzen Überblick über die **wichtigsten Funktionen** von XMLSpy , die dann in den verschiedenen Abschnitten zu den Ansichten der Benutzeroberfläche (Textansicht, Schema/WSDL-Ansicht, Authentic-Ansicht usw.) und in der Benutzerreferenz näher beschrieben sind. Beachten Sie bitte, dass es sich hierbei nicht um eine komplette Liste aller Features handelt. In diesem Abschnitt sollen Sie nur einen ungefähren Eindruck davon erhalten, wofür Sie XMLSpy verwenden können.

Anmerkung: Einige Funktionen von XMLSpy stehen in den Visual Studio .NET-Versionen von XMLSpy nicht zur Verfügung. Nähere Informationen dazu finden Sie unter Unterschiede zwischen der .NET- und der Standalone-Version.

Bearbeiten von XML-Dokumenten in mehreren Editierformaten
Sie können ein XML-Dokument als reinen Text (Textansicht), in einem hierarchischen Tabellenformat (Enhanced Grid-Ansicht) oder in einer grafischen WYSIWYG-Ansicht (Authentic-Ansicht) bearbeiten. Für XML-Schemas und WSDL-Dokumente können Sie auch die Schema/WSDL-Ansicht verwenden, eine grafische Benutzeroberfläche, die das Erstellen von komplexen Schemas und WSDL-Dokumenten viel einfacher macht. Je nach Bedarf können Sie zwischen den verschiedenen Ansichten hin- und herwechseln. Über die Browser-Ansicht können Sie XML-Dokumente, die mit einem XSLT Stylesheet und HTML-Dokumenten verknüpft sind, direkt anzeigen lassen.

Wohlgeformtheitsprüfung und integrierter Validator
Alle XML-Dokumente werden auf ihre Wohlgeformtheit überprüft, wenn Sie die Ansicht wechseln oder die Datei speichern. Außerdem können XML-Dokumente validiert werden, wenn ein Schema (DTD oder XML-Schema) mit dem XML-Dokument verknüpft wurde. Andere Dokumentarten wie z.B. DTDs werden außerdem auch auf Fehler in der Syntax und Struktur überprüft.

Strukturelle Editierfunktionen
In der Textansicht helfen Ihnen Features wie z.B. Zeilennummerierung, Einrücklinien, Lesezeichen und ein- und ausklappbare Elemente schnell und effizient, durch das Dokument zu navigieren.

© 2007 Altova GmbH

Intelligentes Editieren
Wenn ein Schema mit einem XML-Dokument verknüpft wurde, stellt die Autokomplettierungsfunktion der Textansicht eine nützliche Editierhilfe dar. Während der Eingabe erscheinen an der Cursorposition Popup-Menüs, die die Elemente, Attribute und Attributwerte enthalten, die an dieser Stelle eingefügt werden dürfen. Außerdem werden beim Einfügen von Anfangs-Tags automatisch die richtigen End-Tags eingefügt. Attribute, die Sie aus den Popup-Menüs auswählen, werden mit öffnenden und schließenden Anführungszeichen eingefügt. XMLSpy kann außerdem so konfiguriert werden, dass obligatorische Elemente und/oder Attribute beim Einfügen eines Elements automatisch eingefügt werden. Des Weiteren hat jede Ansicht einen Satz von Eingabehilfen, über die Sie Dokumentenkomponenten einfügen oder die Eigenschaften der ausgewählten Komponente im Hauptfenster festlegen können.

Bearbeiten und Verwalten von Schemas
In der grafischen Schema/WSDL-Ansicht lassen sich Schemas schnell und einfach erstellen, da hier keine eingehenden Kenntnisse der Struktur, der Syntax und der Grundlagen des Erstellens von XML-Schemas nötig sind. Sie können auch DTDs erstellen, die auf die korrekte Syntax überprüft werden, und Schemas in DTDs und umgekehrt konvertieren sowie Dokumentation generieren. Zusätzlich dazu haben Sie mit Hilfe des SchemaAgent die Möglichkeit, Schemas von anderen Servern zu verwenden, wodurch Ihnen eine effiziente XML-Umgebung zum Verwalten und Bearbeiten von XML-Schemas zur Verfügung steht.

Integrierter XSLT 1.0- und XSLT 2.0-Prozessor
Die integrierten XSLT 1.0- und XSLT 2.0-Prozessoren entsprechen den Vorgaben der jeweiligen W3C Drafts. Sie gestatten Ihnen das Transformieren von XML-Dokumenten mit Hilfe eines XSLT 1.0- oder XSLT 2.0-Stylesheet direkt von der IDE aus sowie das Debuggen von XSLT Stylesheets mit dem XSLT Debugger.

Integrierter XQuery 1.0-Prozessor
Der integrierte XQuery 1.0-Prozessor entspricht den Standards des XQuery 1.0 W3C Working Draft vom 23. Juli 2004. Sie können damit XQuery-Dokumente direkt von der IDE aus ausführen und debuggen.

Transformation von XML-Dokumenten
XML-Dokumente können direkt in der IDE transformiert werden, entweder mit dem internen (integrierten) XSLT-Prozessor oder jedem beliebigen externen XSLT-Prozessor. Um eine PDF-Datei von der IDE aus zu generieren, können Sie einen externen FO-Prozessor definieren und auch XML nach Angabe des zu verwendenden XSLT Stylesheet mit einem einzigen Klick ins PDF-Format transformieren. Außerdem können von der IDE selbst Parameterwerte an die XSLT-Transformation übergeben werden.

XPath-Evaluierung
Die Funktion "XPath evaluieren" listet für das gewünschte Dokument die Sequence (oder das Node Set) auf, die/das von einem XPath-Ausdruck zurückgegeben wird. Als Context Node können Sie entweder den Dokumenten-Node oder das ausgewählte Element auswählen. Die Funktion "XPath evaluieren" eignet sich vor allem dann, wenn Sie ein XSLT Stylesheet erstellen und einen XPath-Ausdruck dafür evaluieren möchten. Sie können mit Hilfe der Node-Liste in der zurückgegebenen Sequence zu den einzelnen Nodes navigieren.

XSLT Debugger für XSLT 1.0 und XSLT 2.0
Der XSLT Debugger für XSLT 1.0 und XSLT 2.0 entspricht den Standards der entsprechenden

W3C Drafts. Mit Hilfe des XSLT Debuggers kann ein XSLT Stylesheet debuggt werden. Der Debugger führt das zu debuggende XSLT Stylesheet an der XML-Datei aus. Das Resultat wird Schritt für Schritt für jeden einzelnen Schritt der Transformation generiert, wobei Sie den Context Node, die gerade ausgeführte Vorlage und andere Einzelheiten jedes einzelnen Transformationsschritts sehen.

XQuery 1.0 Debugger
Der XQuery 1.0 Debugger entspricht den Standards des XQuery 1.0 W3C Working Draft vom 23. Juli 2004. Er ist dem XSLT Debugger ähnlich und kann zum Debuggen von XQuery-Dokumenten verwendet werden.

XML-Projektverwaltung
Die XMLSpy IDE gestattet Ihnen, miteinander verwandte Dateien in Form einer Baumstruktur in Projekten zu organisieren. Ein Projekt kann aus Schema-Dateien, XML-Dateien, Transformationsdateien und Ausgabedateien bestehen. Die Dateien eines Projekts werden gemeinsam im Projektfenster aufgelistet, sodass Sie bei der Arbeit an einem Projekt in übersichtlicher Form jederzeit zur Verfügung stehen. Zudem können Sie bestimmte Einstellungen für das gesamte Projekt oder für einzelne Ordner festlegen, wie z.B. die Schema-Datei, gegen die Dateien validiert werden sollen oder die zu verwendende Transformationsdatei.

Authentic-Ansicht
Die Authentic-Ansicht ist eine grafische Ansicht des XML-Dokuments in XMLSpy. Über diese Ansicht, können Daten wie über ein Textverarbeitungsprogramm in das XML-Dokument eingegeben werden. Das StyleVision Power Stylesheet, welches in Altova StyleVision erstellt wird, dient dazu, die Formatierung des XML-Dokuments in der Authentic-Ansicht zu definieren und festzulegen, wie die Dateneingabe in Altova StyleVision erfolgt. Beachten Sie bitte, dass die Authentic-Ansicht auch in der derzeit kostenlos erhältlichen Authentic Desktop von Altova zur Verfügung steht.

Datenbankimport
Sie können Datenbankdaten als XML-Datei importieren und anhand der Datenbankstruktur eine XML-Schema-Datei generieren. Es wird der Import der folgenden Datenbanken unterstützt: MS Access, MS SQL Server, Oracle, MySQL, Sybase, IBM DB2.

WSDL und SOAP
Die Schema/WSDL-Ansicht ermöglicht Ihnen das Erstellen und Bearbeiten von WSDL-Dokumenten über eine leicht verständliche grafische Benutzeroberfläche. Auch SOAP Requests lassen sich über die IDE erstellen und debuggen.

Vergleich von XML-Dateien (Differencing)
Mit Hilfe der Vergleichsfunktion in XMLSpy können Sie Unterschiede zwischen zwei XML-Dateien ausfindig machen. Dabei lassen sich die verschiedensten Optionen einstellen, um den Vergleich zu konfigurieren, wie z.B. ob die Reihenfolge von Attributen oder Child-Elementen ignoriert werden soll, ob Entities aufgelöst oder Namespaces ignoriert werden sollen. Mit Hilfe der Vergleichsfunktion lassen sich auch Ordner vergleichen.

Visual Studio .NET Integration
XMLSpy lässt sich in Ihre Visual Studio .NET-Umgebung integrieren. Sie müssen dazu nur eine EXE-Datei von der Altova Website herunterladen und ausführen.

© 2007 Altova GmbH

Eclipse 3.0 Integration
Eclipse 3.0 ist eine Open Source-Umgebung, in der verschiedene Arten von Applikationen in Form von Plug-Ins integriert werden können. XMLSpy für die Eclipse-Plattform ist ein Eclipse 3.0 Plug-In, mit Hilfe dessen Sie die Funktionen einer zuvor installierten XMLSpy Version direkt von der Eclipse 3.0-Plattform aus nutzen können.

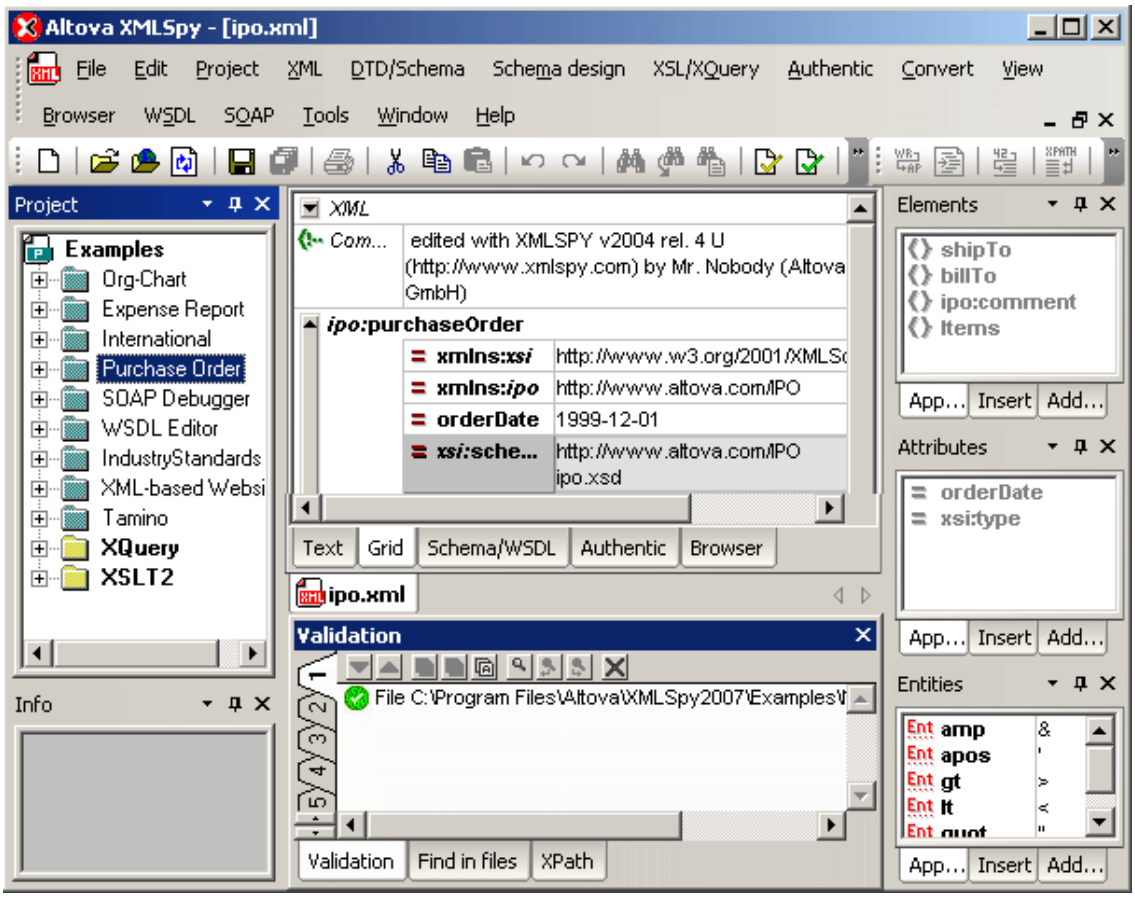
Codegenerierung
Wenn Sie Java-, C++ oder C#-Code verwenden, um Daten in XML-Dateien zu manipulieren, könnten Sie Code mit Klassendefinitionen für das Schema (DTD oder XML-Schema) generieren, auf dem das XML-Dokument basiert. In XMLSpy lässt sich derartiger Code direkt anhand von DTDs und XML-Schemas generieren.

1.3 Benutzeroberfläche

XMLSpy hat eine grafische Benutzeroberfläche (siehe Abb. unten), die in vier Bereiche untergliedert ist:

- das **Projektfenster**, in dem Sie Dateien und Dateigruppen in Projekten organisieren und editieren können; und das **Info-Fenster**, in dem Metainformationen über das derzeit bearbeitete Dokument angezeigt werden.
- das **Hauptfenster**, in dem das Dokument, das Sie gerade bearbeiten, angezeigt wird. Welche Dokumentansichten im Hauptfenster zur Verfügung stehen, hängt von der Art des aktiven Dokuments ab. Sie können jederzeit zwischen den verschiedenen Ansichten wechseln.
- **Eingabehilfe-Fenster**, die je nach Art das angezeigten Dokuments und je nach der Ansicht, die im Hauptfenster ausgewählt ist, variieren können. Die Eingabehilfen dienen zum grafischen Editieren eines Dokuments.

- In den Ausgabefenstern unterhalb des Hauptfensters sehen Sie die Ergebnisse, der Validierung, der Suche in Dateien und der XPath-Auswertungen.



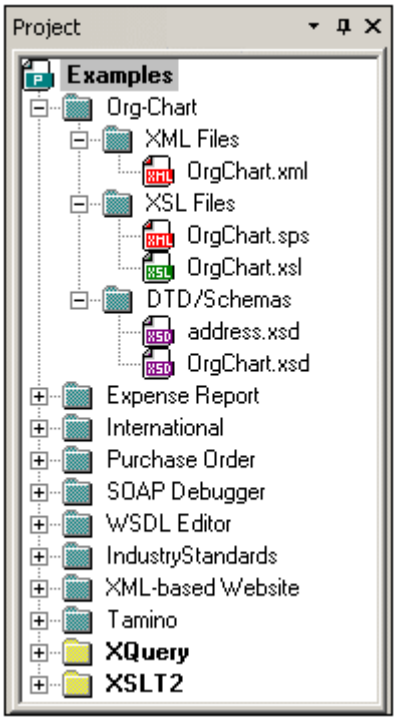
Diese Fenster können unterhalb der Menü- und Symbolleiste angedockt werden (siehe Abbildung oben) oder unterhalb der Menü- und Symbolleiste an beliebiger Stelle angeordnet werden. Position und Größe lassen sich durch Ziehen mit der Maus anpassen. Sie können auch die Befehle im Menü **Fenster** zum Ein- und Ausblenden der Eingabehilfen verwenden.

Dieser Abschnitt enthält eine Einführung in die Grundzüge der Benutzeroberfläche. Ausführliche Beschreibungen zu den verschiedenen Bereichen der Benutzeroberfläche folgen in diesem

Abschnitt.

1.3.1 Projektfenster

XMLSpy verwendet zur Verwaltung mehrerer Dateien oder URLs in XML-Projekten die bekannte Baumstrukturansicht. Sie können Dateien und URLs nach Erweiterung oder beliebigen anderen Kriterien in Ordnern gruppieren, um die Struktur übersichtlicher zu gestalten und Batch-Verarbeitungen zu erleichtern.



Ordner können physischen Verzeichnissen in Ihrem Dateisystem entsprechen oder Sie können Dateierweiterungen für die einzelnen Ordner definieren, nach denen die Dateien geordnet werden. Projektordner sind "semantische" Ordner, die eine logische Dateigruppierung darstellen und **nicht** der hierarchischen Organisation von Dateien auf Ihrer Festplatte entsprechen müssen.

Zuweisen von XSL-Transformationen zu Projektordnern
Sie können jedem Ordner andere XSL-Transformationsparameter zuweisen und sogar ein und dieselbe physische Datei gleichzeitig in mehreren Projektordnern ablegen. Dies eignet sich vor allem, wenn Sie Ihre Daten in einer XML-Datei speichern wollen, aber verschiedene XSL Stylesheets verwenden, um unterschiedliche Ausgabedateien zu erzeugen (z.B. eine separate HTML- und WML-Darstellung).

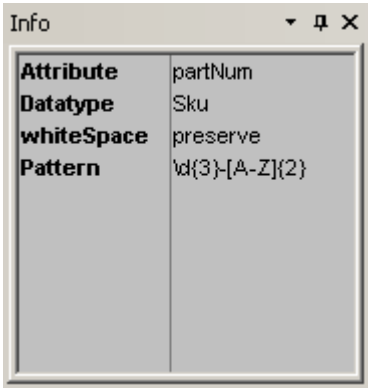
Zuweisen von DTDs / Schemas zu Projektordnern
Sie können verschiedenen Ordnern unterschiedliche DTDs oder Schemas zuweisen. Auf diese Art können Sie eine Datei sowohl gegen eine DTD als auch gegen ein XML-Schema validieren, ohne die Datei selbst zu ändern, was von Vorteil ist, wenn Sie sich gerade in der Phase des Umstiegs von DTDs auf Schemas befinden.

Verwenden Sie zur Projektverwaltung die Befehle im Menü Projekt.

Hinweis: Über die Menüoption **Fenster | Projekt-Fenster** können Sie das Projektfenster ein- oder ausblenden.

1.3.2 Info-Fenster

XMLSpy enthält ein praktisches Informationsfenster, das detaillierte Informationen über das Element bzw. Attribut enthält, auf dem der Cursor gerade positioniert ist.



Diese Informationen stehen in allen Bearbeitungsansichten zur Verfügung und sind besonders in Verbindung mit der `xsd:annotation`-Funktion nützlich.

Hinweis: Sie können das Info-Fenster über die Menüoption **Fenster | Info-Fenster** ein- oder ausblenden.

1.3.3 Hauptfenster

Das Hauptfenster ist der Bereich, in dem alle Dokumente in XMLSpy angezeigt und bearbeitet werden.

Verwalten mehrerer offener Dateien

- Sie können in XMLSpy eine beliebige Anzahl an Dokumenten gleichzeitig geöffnet haben und bearbeiten.
- Jede offene Datei im Hauptfenster wird in einem eigenen Dokumentenfenster angezeigt. Jede dieser Dateien verfügt über ein Register mit dem Namen der Datei, das am unteren Rand des Hauptfensters angezeigt wird.
- Wenn offene Dateien überlappend, nebeneinander angezeigt oder minimiert werden, wird eine Titelleiste mit (i) dem Namen der Datei und (ii) den Standardschaltflächen zum Minimieren, Maximieren und Schließen angezeigt.
- Offene Dateien können maximiert oder minimiert werden, indem Sie in der Titelleiste der offenen Datei auf die jeweilige Schaltfläche zum **Maximieren** bzw. **Minimieren** klicken.
- Wenn Sie eine Datei maximieren, werden auch alle anderen offenen Dateien maximiert.
- Um offene Dateien überlappend oder nebeneinander anzuzeigen, verwenden Sie die Befehle im Menü Fenster.
- Um eine Datei zur aktiven zu machen (um sie zu bearbeiten), klicken Sie auf das Register der Datei oder an eine beliebe Stelle in der jeweiligen Datei. Alternativ dazu können Sie die gewünschte Datei auch im Menü **Fenster** aus der Liste der offenen Dateien im unteren Bereich des Menüs auswählen.
- Sie können offene Dateien auch durch Verwendung von **Strg+Tab** oder **Strg+F6** in der

 © 2007 Altova GmbH

Reihenfolge aktivieren, in der sie geöffnet wurden.

Schneller Zugriff auf Dateibefehle
Um Dateibefehle schnell aufzurufen (wie z.B. Drucken, Schließen, als E-Mail-Anhang versenden), rechtsklicken Sie auf das Dateiregister. Dadurch wird das Kontextmenü mit einer Auswahl an Dateibefehlen geöffnet.

Ansichten im Hauptfenster
XMLSpy bietet verschiedene Ansichten für Ihr XML-Dokument. Dabei handelt es sich entweder um Editier- oder Browser-Ansichten:

- **Textansicht:** Eine Editieransicht mit Syntaxfärbung zur Bearbeitung einer Datei auf Source-Ebene
- **Enhanced Grid-Ansicht:** Zur strukturierten Bearbeitung. Das Dokument wird in Tabellenform dargestellt und kann grafisch bearbeitet werden. Diese Ansicht enthält auch eine eingebettete **Datenbank / Tabellen-Ansicht**, in der sich wiederholende Elemente in Tabellenform dargestellt werden
- **Schema/WSDL-Ansicht:** Zum Anzeigen und Bearbeiten von XML-Schemas und WSDL-Dokumenten
- **Authentic-Ansicht:** Zum Bearbeiten von XML-Dokumenten auf Basis von StyleVision Power Stylesheets
- **Browser-Ansicht:** Eine integrierte Browser-Ansicht, die sowohl CSS- als auch XSL-Stylesheets unterstützt.

Um im Dokumentenfenster zwischen Dokumentenansichten zu wechseln, klicken Sie auf die entsprechende Ansichtsschaltfläche am unteren Rand des Dokumentenfensters oder verwenden Sie die Befehle im Menü Ansicht.

Hinweis: Sie können die Standardansicht (d.h. die Ansicht im Hauptfenster) für bestimmte Dateierweiterungen anpassen. Gehen Sie dazu zu **Extras | Optionen** und nehmen Sie auf den Registern **Dateiarten** und **Ansicht**, die gewünschten Einstellungen vor.

1.3.4 Eingabehilfen

XMLSpy verfügt über intelligente Editierfeatures, mit Hilfe derer Sie schnell gültige XML-Dokumente erstellen können. Diese Features sind in drei palettenartigen Fenstern organisiert, den so genannten Eingabehilfen.

Wenn Sie ein Dokument bearbeiten, sehen Sie in den Eingabehilfen je nach Cursorposition Bearbeitungsoptionen zur Struktur. Die Eingabehilfen beziehen die erforderlichen Informationen aus der zugrundeliegenden DTD, dem XML-Schema und/oder dem StyleVision Power Stylesheet. Wenn Sie z.B. ein XML-Datendokument bearbeiten, werden in den entsprechenden Eingabehilfenfenstern die Elemente, Attribute und Entities angezeigt (sowie dazugehörige Informationen), die an der aktuellen Cursorposition eingefügt werden können.

Die Eingabehilfenfenster haben in Visual Studio .NET ein XMLSpy-Präfix.

Eingabehilfen in verschiedenen Ansichten
Welche Eingabehilfen angezeigt werden, ist abhängig von der gewählten Ansicht. Die verschiedenen Gruppen von Eingabehilfen werden nach den in Ihrem Altova-Produkt verfügbaren Ansichten eingeteilt:

- Textansicht und Grid-Ansicht: Element-, Attribut- und Entities-Eingabehilfe
- Schema Design-Ansicht: Komponenten-Navigator sowie Details- und Facets-Eingabehilfe

- WSDL Design-Ansicht: Übersicht und Details-Eingabehilfe
- Authentic: Element-, Attribut- und Entities-Eingabehilfe

Eine Beschreibung der Eingabehilfen der einzelnen Ansichten finden Sie weiter hinten in der Dokumentation im Abschnitt über die jeweilige Ansicht.

Hinweis: Über die Menüoption **Fenster | Eingabehilfen** können Sie die Eingabehilfen ein- oder ausblenden.

1.3.5 Menü- und Symbolleiste

Menüleiste
Die Menüleiste enthält die verschiedenen Menüs der Applikation. Es gelten die folgenden Konventionen:

- Wenn Befehle in einem Menü in einer Ansicht oder an einer bestimmten Stelle im Dokument **nicht** zur Verfügung stehen, sind sie ausgegraut und deaktiviert.
- Bei Auswahl einiger Menübefehle erscheint ein Untermenü mit einer Liste zusätzlicher Optionen. Solche Menübefehle weisen rechts neben dem Befehlsnamen einen nach rechts weisenden Pfeil auf.
- Bei einigen Menübefehlen wird ein Dialogfeld aufgerufen, in dem Sie weitere Informationen eingeben müssen, damit der Befehl ausgeführt werden kann. Diese Befehle sind durch drei Auslassungspunkte (...) nach dem Namen des Befehls gekennzeichnet.
- Um einen Menübefehl aufzurufen, klicken Sie auf den Menünamen und anschließend auf den Befehl. Enthält ein Menüeintrag ein Untermenü, wird dieses geöffnet, wenn Sie die Maus über den Menüeintrag positionieren. Klicken Sie auf den gewünschten Untermenüeintrag.
- Sie können ein Menü auch durch Drücken der entsprechenden Tastenkombination öffnen. Die Tastenkombination für die einzelnen Menüs lautet **Alt+TASTE**, wobei TASTE für den unterstrichenen Buchstaben im Menünamen steht. Die Tastenkombination für das Menü **Datei** ist z.B. **Alt+a**.
- Die Auswahl eines Menübefehls (also eines Befehls in einem Menü) erfolgt durch das nacheinander-Auswählen (i) des Menüs durch die entsprechende Tastenkombination (siehe voriger Punkt) und (ii) anschließendes Drücken der Tastenkombination für den jeweiligen Befehl (**Alt+TASTE**, wobei TASTE dem unterstrichenen Buchstaben im Befehlsnamen entspricht). Um z.B. eine neue Datei zu erstellen, (**Datei | Neu**), drücken Sie **Alt+a** und dann **Alt+N**.
- Einige Menübefehle können **direkt** durch Drücken eines speziellen **Tastenkürzels** oder einer Tastenkombination (**Strg+TASTE**) aufgerufen werden. Bei Befehlen, denen ein Tastenkürzel zugewiesen wurde, wird rechts vom Befehl das Tastenkürzel oder die Tastenkombination angezeigt. So können Sie z.B. eine neue Datei mit der Tastenkombination **Strg+N** erstellen und eine XML-Datei durch Drücken von **F8** validieren.

Symbolleiste
Die Symbolleiste enthält Schaltflächen zum Aufrufen von Menübefehlen. Wenn Sie den Mauszeiger über die Schaltfläche halten, wird der Name des Befehls angezeigt. Um den Befehl auszuführen, klicken Sie auf die Schaltfläche.

Die Schaltflächen der Symbolleiste sind in Gruppen angeordnet. Im Dialogfeld **Extras | Anpassen | Symbolleisten** können Sie einstellen, welche Symbolleistengruppen angezeigt werden sollen. Sie können Symbolleistengruppen auf der GUI auch an eine andere Stelle ziehen, indem Sie darauf klicken und mit dem Ziehpunkt an die gewünschte Stelle verschieben.

© 2007 Altova GmbH

1.3.6 Ausgabefenster

Die Ausgabefenster befinden sich standardmäßig unterhalb des Hauptfensters. Dies sind die drei Ausgabefenster:

- **Validierung:** Hier werden die Ergebnisse der Dokumentvalidierung im Detail angezeigt.
- **In Dateien suchen:** Hier können Sie die Optionen für das Suchen und Ersetzen in mehreren Dateien definieren. Sie können insgesamt neun verschiedene Suchen durchführen, die alle auf separaten Registern angezeigt werden. Mit Hilfe von Steuerelementen in der Symbolleiste des Fensters "In Dateien suchen" können Sie durch die Ergebnisse der Suche navigieren und Einträge in Dateien finden, indem Sie im Dialogfeld "In Dateien suchen" auf den jeweiligen Eintrag klicken.
- **Auswertung von XPath-Ausdrücken:** In diesem Fenster können Sie Ausdrücke eingeben. Das Ergebnis des Ausdrucks wird im Fenster angezeigt. Dieses Fenster ist v.a. deshalb sehr nützlich, weil Sie sowohl das XML-Dokument als auch das Ergebnis der Auswertung in derselben Ansicht sehen.

Alle Ausgabefenster werden in dieser Dokumentation im Abschnitt Text-Ansicht im Detail angezeigt.

2 Tutorials

Dieser Abschnitt enthält die folgenden Tutorials:

- XMLSpy Tutorial: Ein Tutorial, in dem Schritt für Schritt beschrieben wird, wie ein XML-Schema erstellt wird, bzw. eine XML-Datei erstellt, bearbeitet und transformiert wird, Datenbanken in XMLSpy für XML verwendet werden und zusammengehörige Dateien in XMLSpy-Projekten organisiert werden können.
- Authentic-Ansicht Tutorial: Ein Tutorial, in dem die einzigartige Benutzeroberfläche der Altova Authentic-Ansicht und deren Features beschrieben werden. In der Authentic -Ansicht können Sie XML-Dokumente über eine graphische Benutzeroberfläche bearbeiten, ohne XML-Kenntnisse zu benötigen oder die Struktur des zugrunde liegenden Schemas zu kennen.
- WSDL Tutorial: ein Tutorial, in dem die Grundzüge der WSDL Design-Ansicht, in der Sie WSDL-Dokumente schnell und einfach erstellen können, erklärt werden.

© 2007 Altova GmbH

2.1 XMLSpy Tutorial

Mit der Ihnen vorliegenden Anleitung erhalten Sie einen umfassenden Einblick in XML. Schritt für Schritt werden Sie mit den Funktionen von XMLSpy vertraut.

Mit dieser Anleitung lernen Sie folgendes:

- Generieren von **einfachen Schemas**
- **Verallgemeinern von Schemas** durch simple und complex types
- Erstellen von Schema-**Dokumentation**
- Erstellen von **XML-Dokumenten** basierend auf einer Schema-Datei
- Kopieren von XML-Daten in andere Anwendungen (z.B. Excel), sowie Wiedereinfügen in XMLSpy
- **Validierung** des XMS Dokumentes gegen das zugewiesene Schema
- **Aktualisierung** von Schema-Einstellungen während der Bearbeitung des XML Dokumentes
- **Transformieren** des XML-Dokumentes in HTML durch Verwendung von XSLT und Anzeigen der Ergebnisse in der Browser-Ansicht
- **Import** und **Export** von Datenbankeneinträgen von und nach XMLSpy
- Erstellen eines **Schemas** anhand von einer MS Access-Datenbank
- Erstellen eines XMLSpy-**Projektes** zum Organisieren Ihrer XML-Dokumente

Installation und Konfiguration
In dieser Anleitung gehen wir davon aus, dass Sie XMLSpy erfolgreich auf Ihrem PC installiert haben, dass Sie einen kostenlosen Evaluierungs-Key erhalten haben oder dass Sie sich als User bereits registriert haben. Die Evaluierungsversion von XMLSpy ist voll funktionsfähig. Allerdings ist diese auf dreißig Tage begrenzt. Sie können danach eine reguläre Lizenz über unseren sicheren Webserver oder über einen Vertriebspartner erwerben.

Anleitung: Beispieldateien
Beispieldateien, die in der vorliegenden Anleitung angesprochen werden, finden Sie im Applikationsordner unter:

```
XMLSpy2007\Examples\Tutorial
```

Im Verzeichnis `Examples` finden Sie viele XML-Dateien, mit denen Sie experimentieren können. Im Verzeichnis `Tutorial` sind alle Dateien abgelegt, die in dieser Anleitung verwendet werden.

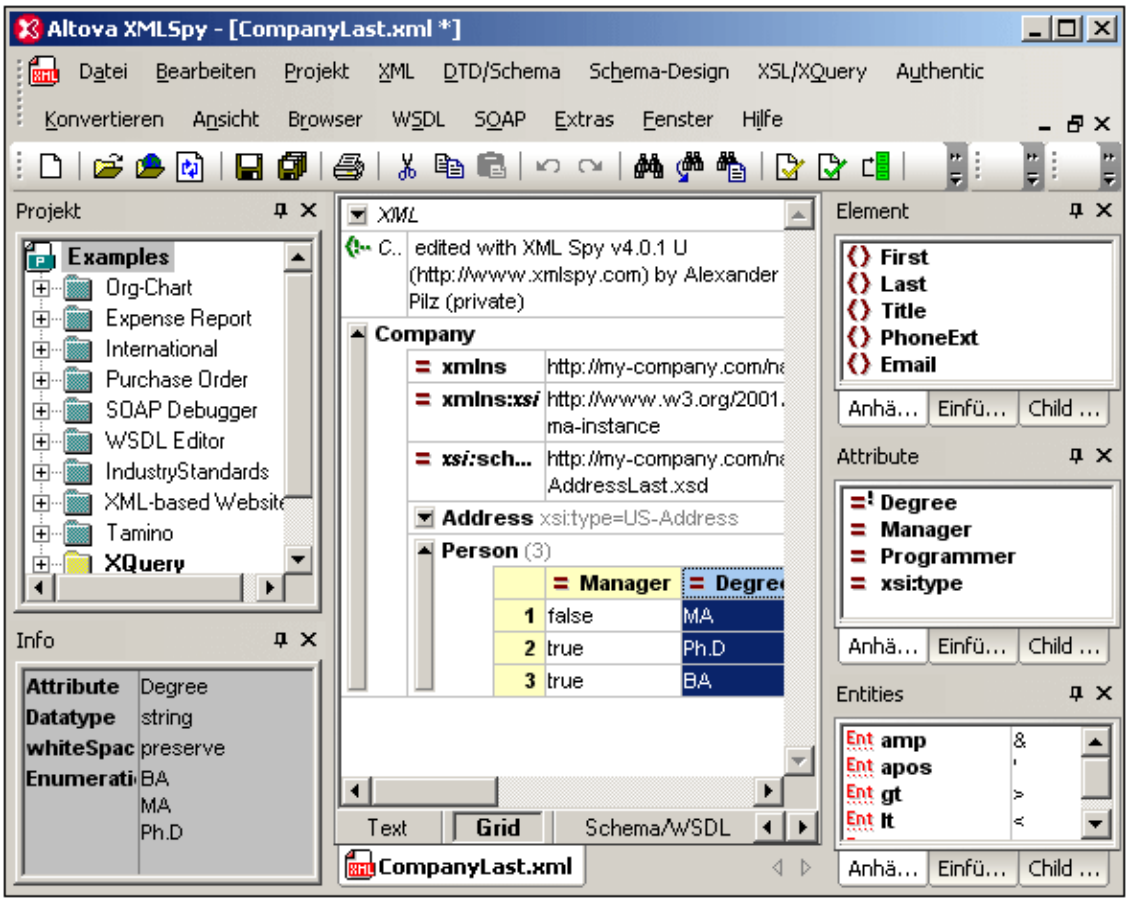
Das Verzeichnis `Template` **(Vorlagen)** enthält sämtliche XML-Vorlagen, die verwendet werden, wenn Sie die Menüoption **Datei | Neu** aufrufen. Diese Dateien enthalten alle nötigen Daten (Namespace und XML-Deklarationen) die Sie benötigen, um mit dem XML-Dokument arbeiten zu können.

2.1.1 Die XMLSpy-Benutzeroberfläche

Die Oberfläche von XMLSpy ist vertikal in drei Bereiche gegliedert. Im mittleren Bereich stehen verschiedene Ansichten für Ihr Dokument bereit. Die beiden seitlichen Bereiche enthalten Informationen, Editierhilfen und Funktionen zur Dateiverwaltung.

- Das **Projekt-** und das **Info-Fenster** befinden sich auf der linken Seite.
- Der mittlere Teil besteht aus dem **Hauptfenster**. Hier bearbeiten Sie Ihr XML-Dokument. Sie können zwischen verschiedenen Ansichten wählen: Textansicht, Grid-Ansicht, Schema/WSDL Design-Ansicht, Authentic-Ansicht und Browser-Ansicht. Diese Ansichten werden in den entsprechenden Abschnitten dazu näher beschrieben.

- Auf der rechten Seite des Bildschirmes befinden sich drei **Eingabenhilfen**-Fenster. Hier können Sie Elemente, Attribute oder Entities einfügen oder anhängen. Die verfügbaren Einträge ändern sich je nachdem, welche Auswahl in der Grid-Ansicht vorgenommen wurde, bzw. je nach Positionierung des Cursors in der XML-Datei.



Die Einzelheiten der Benutzeroberfläche werden im Laufe des Handbuchs näher beschrieben. Beachten Sie, dass die Oberfläche sich dynamisch je nach gewählter Ansicht ändert und je nachdem, welches Dokument im Hauptfenster aktiv ist.

2.1.2 Erstellen eines einfachen XML-Schemas

Ein XML-Schema beschreibt die Struktur eines XML-Dokuments. Ein XML-Dokument kann gegen ein XML-Schema validiert werden, um zu überprüfen, ob es mit den im Schema definierten Vorgaben übereinstimmt und somit gültig ist. Falls dies nicht der Fall ist, ist es ungültig. Mit Hilfe von Schemas kann der Verfasser eines Dokuments die Struktur und den zulässigen Inhalt eines XML-Dokuments definieren und überprüfen, ob das XML-Dokument gültig ist.

Ein XML-Schema hat eine komplexe Struktur und Syntax. Da es sich bei einem XML-Schema selbst um ein XML-Dokument handelt, muss es den Regeln der XML-Schema-Spezifikation entsprechen. In XMLSpy können Sie in der Schema/WSDL Design-Ansicht mit Hilfe von Drag & Drop-Techniken gültige XML-Schemas auf einfache Art erstellen. Das von Ihnen erzeugte XML-Schema-Dokument kann auch in der Text- und der Grid-Ansicht bearbeitet werden, viel einfacher lässt sich dies jedoch in der Schema/WSDL-Ansicht bewerkstelligen.

In diesem Abschnitt lernen Sie, wie man ein XML-Schemas in der Schema/WSDL

Design-Ansicht bearbeitet. Insbesondere wird auf folgende Punkte eingegangen:

- Erstellen einer neuen Schema-Datei
- Definieren von Namespaces für das Schema
- Definieren eines einfachen Content Model
- Hinzufügen von Elementen zum Content Model über Kontextmenüs und mittels Drag & Drop
- Konfiguration der Content Model-Ansicht

Nachdem Sie mit der Erstellung des Schema-Grundgerüsts fertig sind, können Sie zum nächsten Abschnitt des Tutorials übergehen, in dem Sie erfahren, wie man mit komplexeren XML-Schema-Funktionen in XMLSpy arbeitet. Auf diesen Abschnitt folgt ein Abschnitt über Schema-Navigation und Dokumentation in XMLSpy.

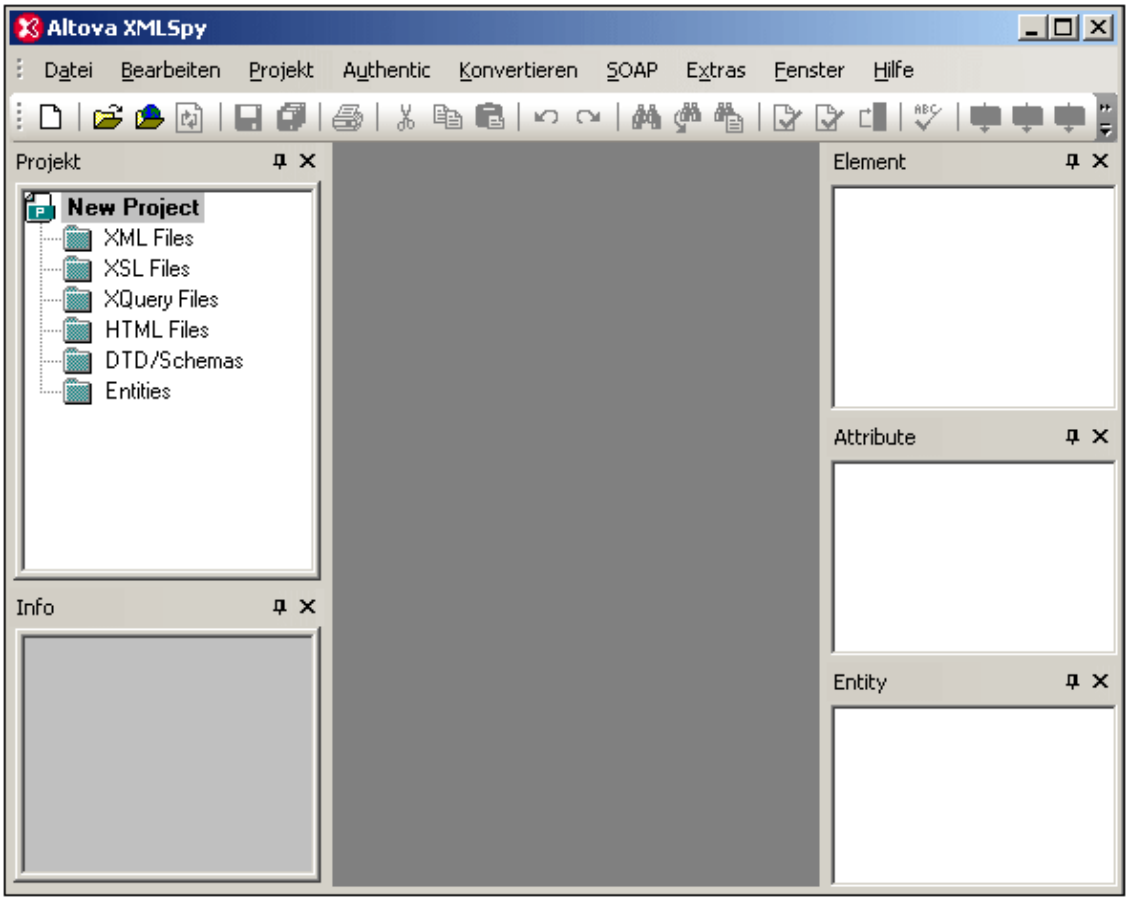
In diesem Abschnitt verwendete Befehle
In diesem Abschnitt arbeiten Sie ausschließlich in der Schema/WSDL-Ansicht. Dabei werden die folgenden Befehle verwendet:

Diagramm anzeigen (oder Content Model-Ansicht anzeigen). Dieses Symbol befindet sich links von allen globalen Komponenten in der Schema-Übersicht. Wenn Sie auf das Symbol klicken, wird das Content Model der dazugehörigen globalen Komponente angezeigt.

Anlegen einer neuen XML-Schema-Datei

Um in XMLSpy eine neue XML-Schema-Datei anzulegen, müssen Sie zuerst XMLSpy starten und dann ein neues XML-Schema (. `xsd`)-Dokument erstellen.

Starten von XMLSpy
Doppelklicken Sie zum Starten von XMLSpy auf das XMLSpy-Symbol auf Ihrem Desktop oder rufen Sie XMLSpy über **Start | Programme** auf. XMLSpy wird mit einem leeren Hauptfenster gestartet, in dem kein Dokument geöffnet ist.

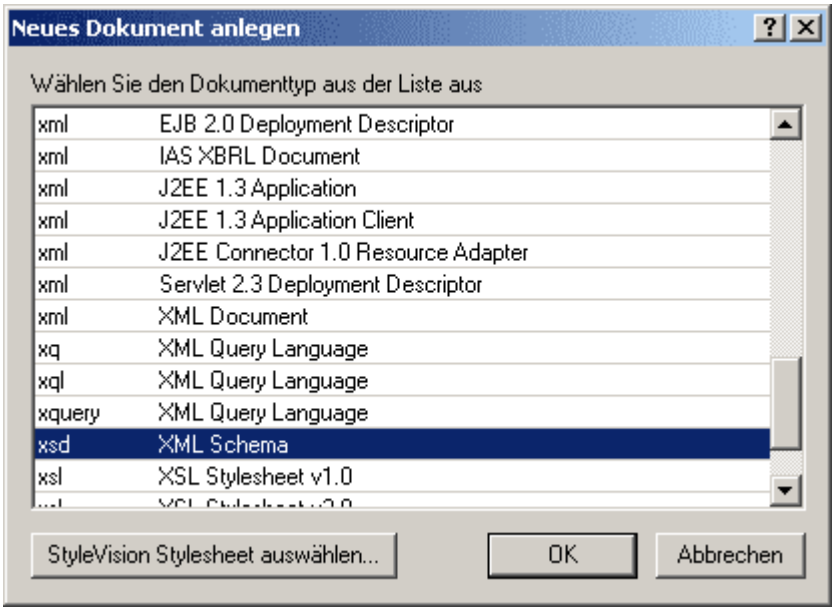


Beachten Sie, dass sich die Benutzeroberfläche in drei Teile gliedert: (i) das Projekt- und das Info-Fenster auf der linke Seite; (ii) das Hauptfenster in der Mitte; und (iii) die Eingabehilfen auf der rechten Seite.

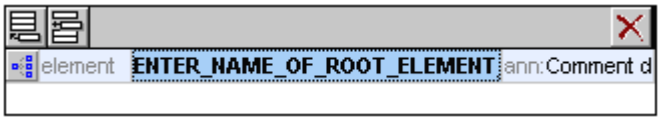
© 2007 Altova GmbH

So legen Sie eine neue Schema-Datei an:

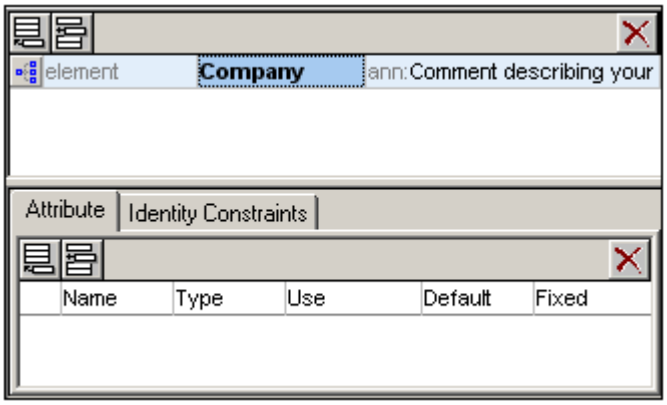
1. Wählen Sie die Menüoption **Datei | Neu**. Das Dialogfeld "Neues Dokument anlegen" wird angezeigt.



2. Wählen Sie im Dialogfeld den Eintrag `. xsd W3C Schema .` Bestätigen Sie mit OK. Eine leere Schema-Datei wird im Hauptfenster in der Schema/WSDL Design-Ansicht angezeigt. Sie werden aufgefordert, den Namen des Root-Elements anzugeben.



3. Klicken Sie in das markierte Feld, und geben Sie `Company` ein. Bestätigen Sie mit der Eingabetaste. `Company` ist somit das "Root" Element dieses Schemas und gleichzeitig ein globales Element. Die im Hauptfenster angezeigte Ansicht (*siehe Abbildung unten*) nennt sich Schema-Übersicht und enthält eine Liste aller globalen Komponenten im oberen Bereich Fenster; die Attribute und Identity Constraints der ausgewählten Komponente werden im unteren Fenster angezeigt. (Sie können das Content Model einzelner globaler Komponenten ansehen und bearbeiten, indem Sie auf das Symbol "Diagramm anzeigen" klicken, das links von der globalen Komponente zu sehen ist.)



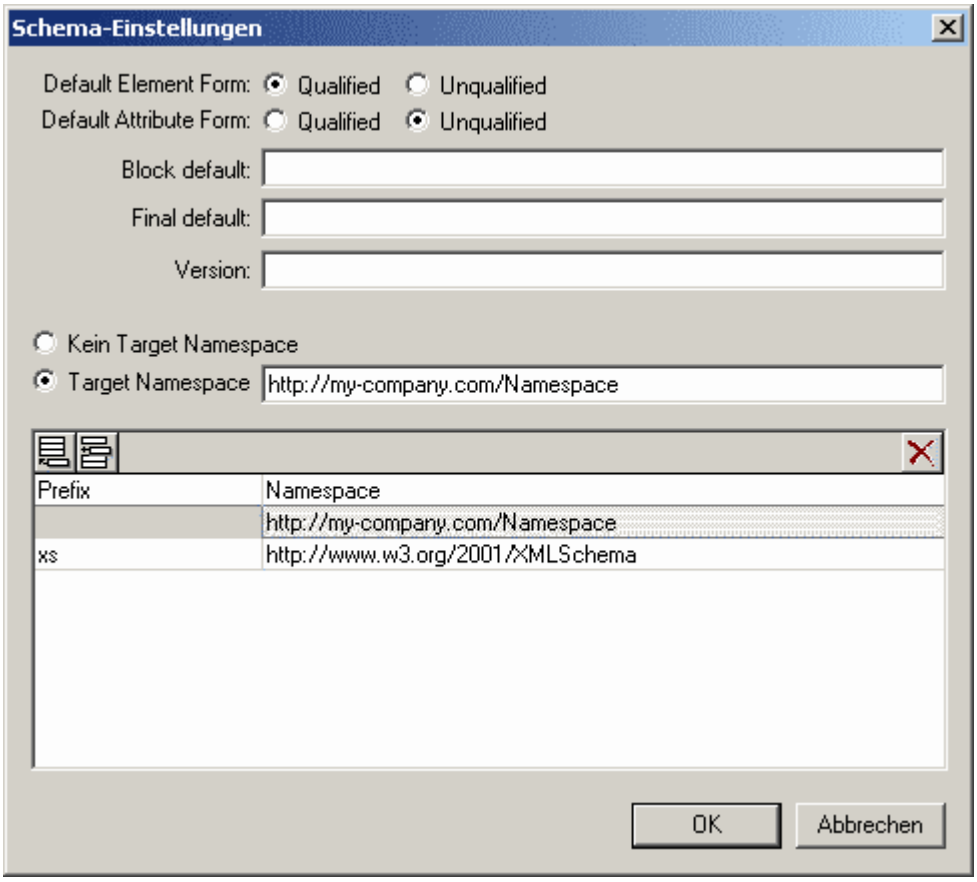
4. Geben Sie im Annotationsfeld (`ann`) des Elements `Company` die Beschreibung für das Element ein, in diesem Fall `Root element`.
5. Klicken Sie auf die Menüoption **Datei | Speichern** und benennen Sie Ihr Schema (z.B. `AddressFirst.xsd`).

Definieren von Namespaces

XML Namespaces spielen eine wichtige Rolle in XML-Schemas und XML-Dokumenten. Ein XML-Schema muss den XML-Schema-Namespace referenzieren und kann optional auch einen Target Namespace für das XML-Instanzdokument definieren. Als Schema-Autor müssen Sie festlegen, wie Sie diese beiden Namespaces definieren (im Grunde mit Hilfe von Präfixen).

Sie werden nun im XML-Schema, das Sie erstellen, einen Target Namespace für das XML-Instanzdokument definieren. (Die erforderliche Referenz zum XML Schema Namespace wird von XMLSpy automatisch erstellt, wenn Sie ein neues XML Schema-Dokument anlegen.) So erstellen Sie einen Target Namespace:

1. Wählen Sie die Menüoption **Schema-Design | Schema-Einstellungen**, um das Dialogfeld "Schema-Einstellungen" aufzurufen.



2. Klicken Sie auf das Target Namespace-Optionsfeld und geben Sie `http://my-company.com/Namespace` ein. In XMLSpy wird der von Ihnen als Target Namespace eingegebene Namespace als Default Namespace des XML-Schema-Dokuments erstellt und in der Liste der Namespaces im unteren Bereich des Dialogfelds angezeigt.
3. Bestätigen Sie mit OK.

© 2007 Altova GmbH

Bitte beachten Sie:

- Der XML Schema Namespace wird von XMLSpy automatisch erstellt und erhält das Präfix `xs:` .
- Wenn Sie das XML-Instanzdokument erstellen, muss es den im XML-Schema für das XML-Dokument definierten Target Namespace haben, um gültig zu sein.

Definieren eines Content Model

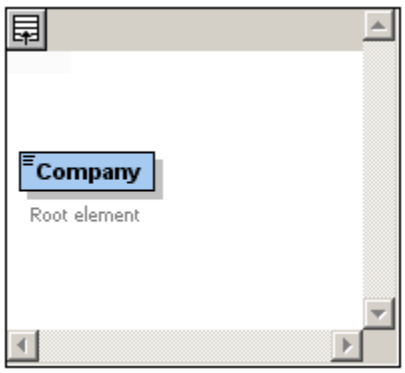
Sie haben in der Schema-Übersicht bereits ein globales Element mit dem Namen `Company` erstellt. Dieses Element soll ein `Address`-Element und eine unbegrenzte Anzahl an `Person` -Elementen enthalten, nämlich das Content Model. Globale Elemente, die Content Models haben können, sind Elemente, complexTypes und Elementgruppen.

In XMLSpy wird das Content Model einer globalen Komponente in der Content Model-Ansicht der Schema/WSDL-Ansicht angezeigt. Um das Content Model einer globalen Komponente anzeigen und editieren zu können, klicken Sie auf das Symbol "Diagramm anzeigen" links neben der globalen Komponente.

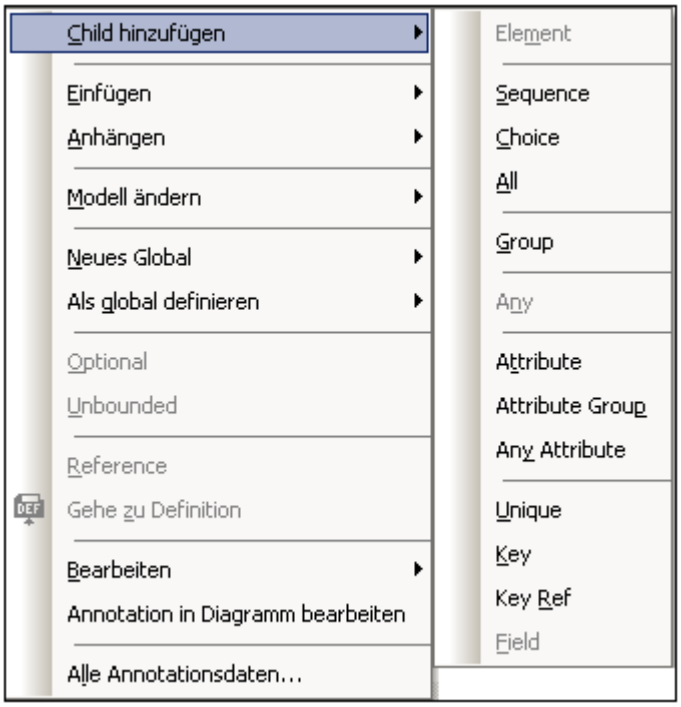
In diesem Abschnitt werden Sie das Content Model des Elements `Company` erstellen.

Erstellen eines einfachen Content Model
So erstellen Sie das Content Model des Elements `Company`:

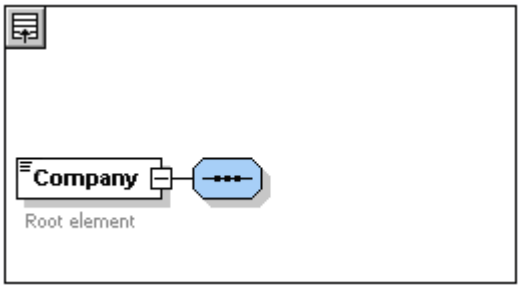
1. Klicken Sie in der Schema-Übersicht auf das Symbol "Diagramm anzeigen" links neben dem Element `Company`. Hierdurch wird das Content Model des Elements `Company` angezeigt, das derzeit noch leer ist. (Alternativ dazu können Sie auch auf den Eintrag `Company` im Komponentenfenster doppelklicken, um das Content Model anzuzeigen.)



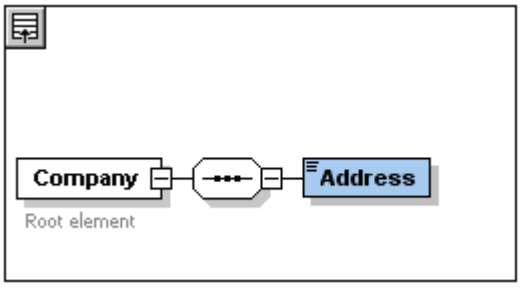
2. Ein Content Model besteht aus Kompositoren und Komponenten. Die Kompositoren definieren die Beziehung zwischen zwei Komponenten. Sie müssen nun im `Company` Content Model einen Child-Kompositor zum Element `Company` hinzufügen, um ein Child-Element hinzufügen zu können. Rechtsklicken Sie dazu auf das Element "Company" und wählen Sie im Kontextmenü die Option **Child hinzufügen | Sequence**. (Sequence, Choice und All sind die drei Kompositoren, die in einem Content Model verwendet werden können.)



Daraufhin wird der Sequence-Kompositor eingefügt, der definiert, dass die Komponenten, die darauf folgen in der angegebenen Reihenfolge erscheinen müssen.

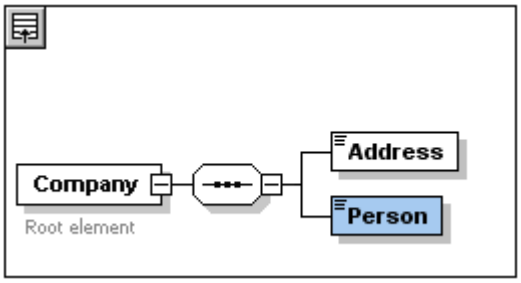


3. Rechtsklicken Sie auf den Sequence-Kompositor und wählen Sie **Child hinzufügen | Element**. Daraufhin wird ein unbenanntes Element hinzugefügt.
4. Benennen Sie das Feld `Address` und bestätigen Sie mit der Eingabetaste.



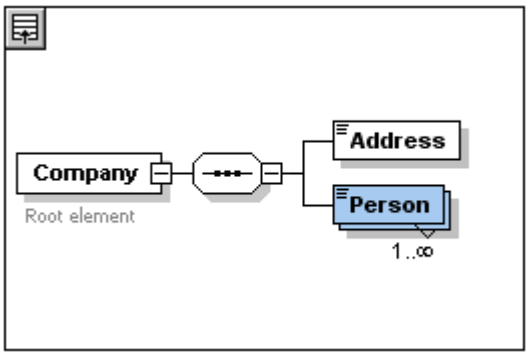
5. Rechtsklicken Sie erneut auf den Sequence-Kompositor und wählen Sie **Child hinzufügen | Element**, benennen Sie die neue Elementkomponente `Person`.

© 2007 Altova GmbH



Sie haben nun ein Schema mit einem Eintrag für die Firmenadresse und einem Eintrag für einen Angestellten (`Person`) definiert. Wir müssen nun die Anzahl der `Person` -Elemente erhöhen.

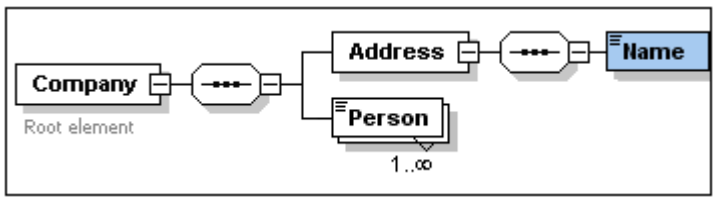
6. Klicken Sie mit der rechten Maustaste auf das Element `Person` und wählen Sie **Unbounded** aus dem Kontextmenü. Unter dem Element `Person` wird nun die Anzahl der zulässigen Instanzen angezeigt: 1 bis unendlich.



Sie können alternativ dazu in der Details-Eingabehilfe die Felder `minOcc` und `maxOcc` direkt editieren und die Anzahl der zulässigen Instanzen angeben, in diesem Fall 1 bzw. unbounded.

Hinzufügen von zusätzlichen Ebenen zur Struktur des Content Model
Bisher enthält das Content Model, das Sie erstellt haben, eine Ebene: eine Child-Ebene für das Element `company`, das die Elemente `Address` und `Person` enthält. Nun wollen wir den Inhalt des Elements `Address` definieren, und zwar die Subelemente `Name`, `Street` und `City`. Dies ist eine zweite Ebene. Wir müssen also zum Element `Address` einen Child-Kompositor hinzufügen und anschließend die Elementkomponenten selbst. Gehen Sie dabei folgendermaßen vor:

1. Klicken Sie mit der rechten Maustaste auf das Element `Address`, um das Kontextmenü zu öffnen, und wählen Sie **Child hinzufügen | Sequence**.
2. Klicken Sie mit der rechten Maustaste auf den **Sequence-**Kompositor und wählen Sie **Child hinzufügen | Element**. Benennen Sie die neu erstellte Elementkomponente mit "Name".



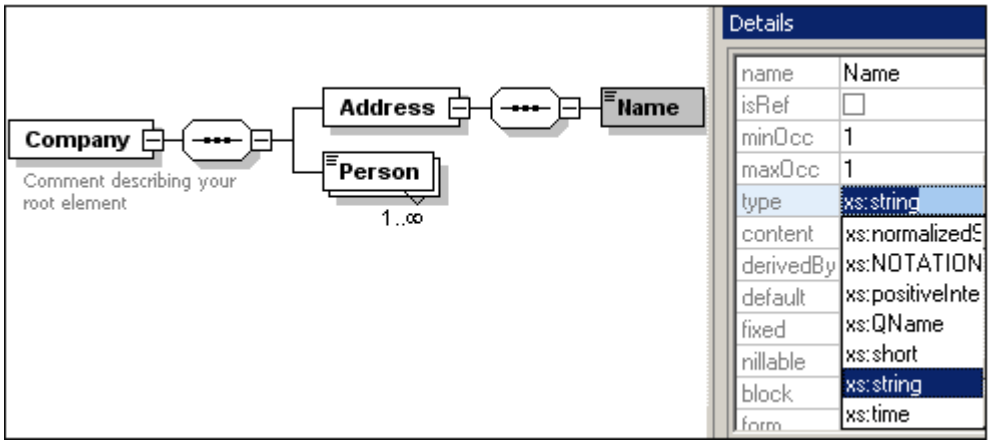
ComplexTypes, simpleTypes und XML Schema-Datentypen
Bisher haben wir noch keinen Elementtyp explizit definiert. Klicken Sie auf das Register **Text**

um die Textansicht für Ihr Schema anzuzeigen (*siehe Liste unten*). Sie werden feststellen, dass jedes Mal, wenn ein Sequence-Kompositor eingefügt wurde, das Element `xs:sequence` im Element `xs:complexType` eingefügt wurde. Kurz gesagt, die Elemente `Company` und `Address` sind ComplexTypes, da sie Child-Elemente enthalten. Ein ComplexType ist ein Element, das Attribute oder Elemente enthält.

```
<xs:element name="Company">
  <xs:annotation>
    <xs:documentation>Root element</xs:documentation>
  </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>
```

SimpleType-Elemente dagegen enthalten nur Text und haben keine Attribute. Beim Text kann es sich um Strings, Daten, Zahlen usw. handeln. Wir möchten, dass das Child-Element `Name` von `Address` nur Text enthält. Es ist ein SimpleType, dessen Textinhalt wir durch Restriktion auf einen String einschränken möchten. Dies lässt sich durch Verwendung des XML-Schema-Datentyps `xs:string` bewerkstelligen. Um das Element `Name` als diesen Datentyp zu definieren, gehen Sie folgendermaßen vor.

1. Klicken Sie auf das Register **Schema/WSDL** um zur Schema/WSLD-Ansicht zurückzukehren.
2. Klicken Sie auf `Name`, um das Element auszuwählen.
3. Klicken Sie in der Details-Eingabehilfe auf das Listenfeld `type` in der mittleren Eingabehilfe und wählen Sie `xs:string` aus der Auswahlliste.



Beachten Sie, dass sowohl `minOcc` als auch `maxOcc` den Wert 1 hat, d.h. dieses Element hat nur eine Instanz.

In der Textdarstellung sieht das Element `Name` folgendermaßen aus:

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

Bitte beachten Sie: Ein einfaches Type-Element kann jedem beliebigen von mehreren

© 2007 Altova GmbH

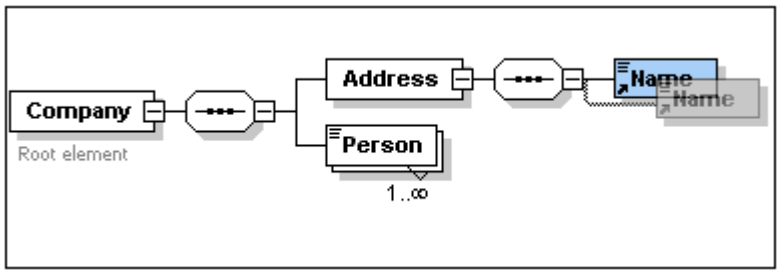
XML-Schema-Datentypen angehören. In all diesen Fällen erscheint das Symbol für Textinhalt nur im Elementfeld.

Hinzufügen von Elementen mit Drag & Drop

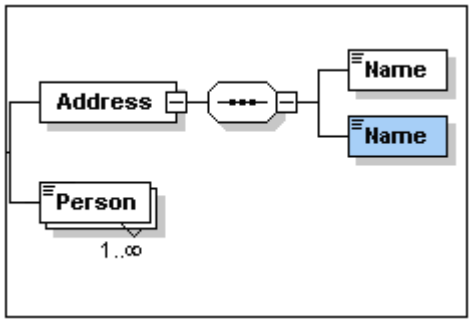
Sie haben Elemente bisher über das Kontextmenü hinzugefügt, das erscheint, wenn Sie mit der rechten Maustaste auf ein Element oder einen Kompositor klicken. Schneller lassen sich Elemente aber mit Drag & Drop erstellen. In diesem Abschnitt lernen Sie, wie man mit Drag & Drop mehr Elemente zur Definition des Elements Address hinzufügt und so diese Definition fertig stellt.

So stellen Sie de Definition des Elements Address mit Hilfe von Drag & Drop fertig:

1. Klicken Sie auf das Element Name des Elements Address, halten Sie die Maustaste sowie die STRG-Taste gedrückt und ziehen Sie die Maus leicht nach unten. Die Markierung mit einem +Zeichen erscheint. Hierdurch wird angezeigt, dass das Element kopiert werden kann. Es erscheint eine Kopie des Elements sowie eine Konnektorlinie, sodass Sie sehen, wo das Element erstellt wird.

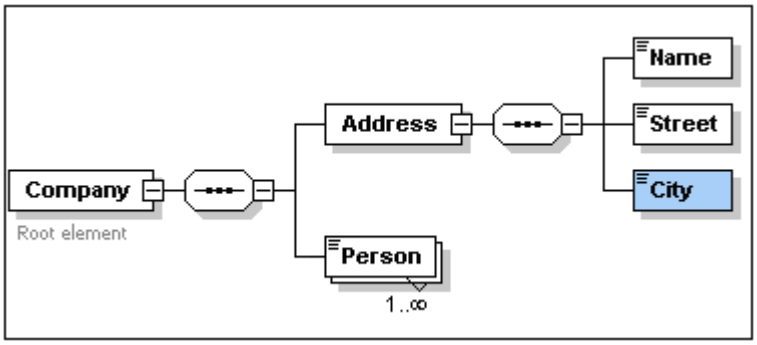


2. Lassen Sie nun die Maustaste und die STRG-Taste los um das neue Element in der Address-Sequenz anzulegen. Wurde das Element an der falschen Stelle angelegt, so ziehen Sie es einfach mit der Maus unter das Element Name.



3. Doppelklicken Sie in das Elementfeld und geben Sie Street ein, um den Namen des Elements zu ändern.
4. Wiederholen Sie obige Schritte um ein weiteres Element zu erzeugen. Benennen Sie dieses Feld um und geben Sie ihm den Namen City.

Das Content Model sollte nun folgenden Inhalt haben:



Das Element Address enthält eine Sequence mit einem Element Name, einem Element Street und einem Element City, und zwar in der obigen Reihenfolge.

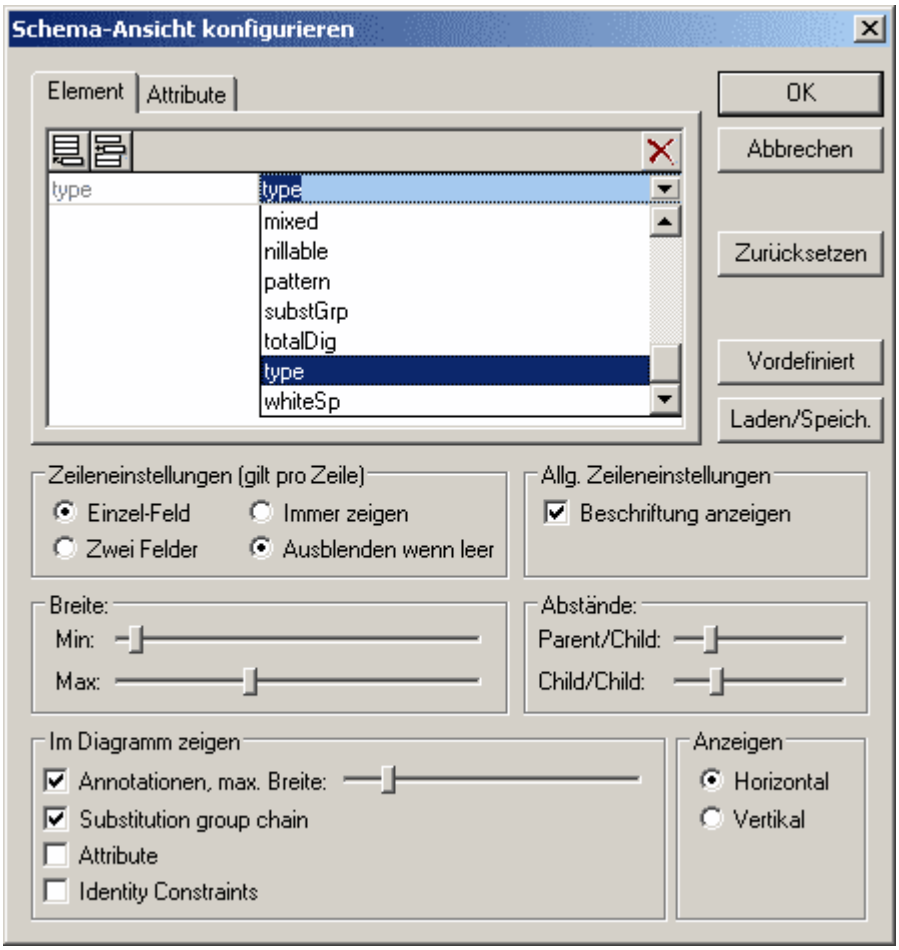
Konfigurieren der Content Model-Ansicht

Sie können jetzt das Aussehen der Content Model-Ansicht konfigurieren. Wir wollen die Content Model-Ansicht so konfigurieren, dass der Elementtyp für jedes einzelne Element angezeigt wird.

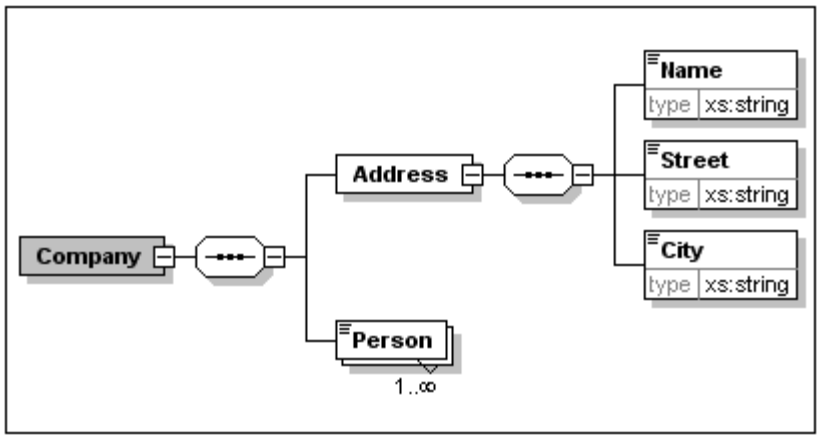
So konfigurieren Sie die Content Model-Ansicht:

1. Gehen Sie zur Content Model-Ansicht einer Komponente (klicken Sie auf das Symbol Content Model-Ansicht), um den Befehl "Ansicht konfigurieren" zu aktivieren.
2. Wählen Sie die Menüoption Schema-Design | Ansicht konfigurieren, um das Dialogfeld "Schema-Ansicht konfigurieren" aufzurufen.

© 2007 Altova GmbH

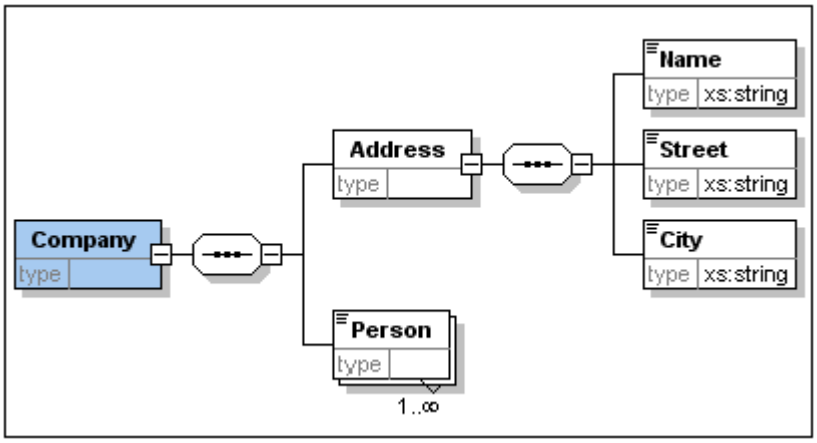


3. Klicken Sie im Element/Attribut-Fenster auf dem Register "Element" auf das Symbol Anhängen um eine Zeile zur Beschreibung einer Eigenschaft zu jedem Elementfeld hinzuzufügen.
4. Wählen Sie type aus der Auswahlliste (oder doppelklicken Sie in die Zeile und tippen Sie "type"). Dadurch wird der Datentyp jedes Elements in der Content Model-Ansicht angezeigt.
5. Wählen Sie im Bereich "Zeileneinstellungen (Gilt pro Zeile)" die Option "Ausblenden wenn leer", damit die Beschreibung des Datentyps im Elementfeld ausgeblendet wird, wenn das Element keinen Datentyp hat (z.B. wenn das Element ein ComplexType ist)



Beachten Sie, dass die Beschreibungszeile für den Typ bei den Elementen Name, Street und City aufscheint, da es sich dabei um SimpleTypes vom Typ xs:string handelt, nicht aber bei den ComplexType-Elementen, da die Option "Ausblenden wenn leer" aktiv ist.

6. Aktivieren Sie im Bereich "Zeileneinstellungen (Gilt pro Zeile)" die Option "Immer zeigen".
7. Klicken Sie auf OK, um die Änderungen zu speichern.



Beachten Sie, dass die Beschreibungszeile für den Datentyp immer angezeigt wird - selbst in Elementfeldern von ComplexTypes, in denen keine Werte in dieser Zeile angezeigt werden.

Bitte beachten Sie:

- Die Eigenschaftsbeschreibungszeilen können editiert werden. Werte, die Sie dort eingeben, werden Teil der Elementdefinition.
- Die Einstellungen, die Sie im Dialogfeld "Schema-Ansicht konfigurieren" vornehmen, gelten für die Ausgabe der Schema-Dokumentation ebenso wie für die Druckausgabe.

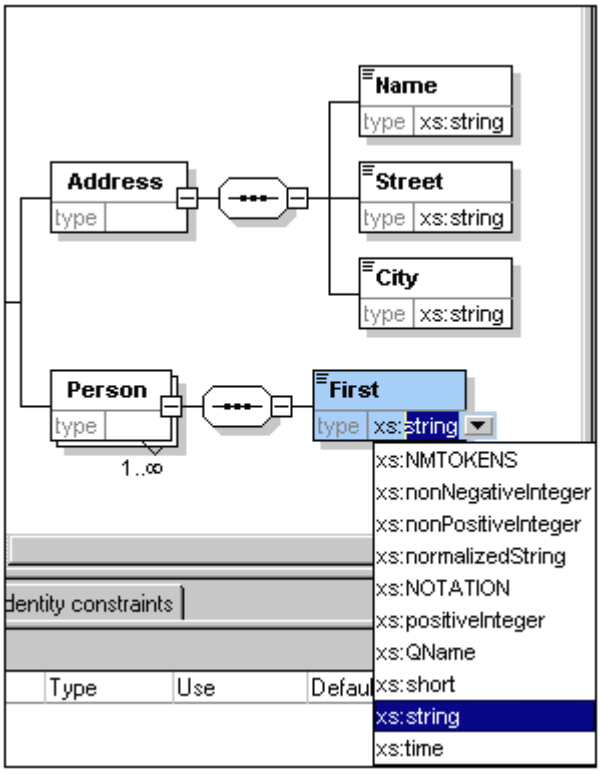
Fertigstellen des Schemas

Nachdem Sie nun den Inhalt des Elements Address definiert haben, müssen Sie den Inhalt des Elements Person definieren. Das Element Person soll die folgenden Child-Elemente - alle vom Typ SimpleType - enthalten: First, Last, Title, PhoneExt und Email. Alle diese Elemente sind obligatorisch mit Ausnahme des Elements Title (das optional ist), und alle Elemente müssen in der oben angegebenen Reihenfolge aufscheinen. Alle Elemente sollen vom Datentyp xs:string sein, mit Ausnahme von PhoneExt, das vom Datentyp xs:integer sein muss und maximal 2 Stellen haben darf.

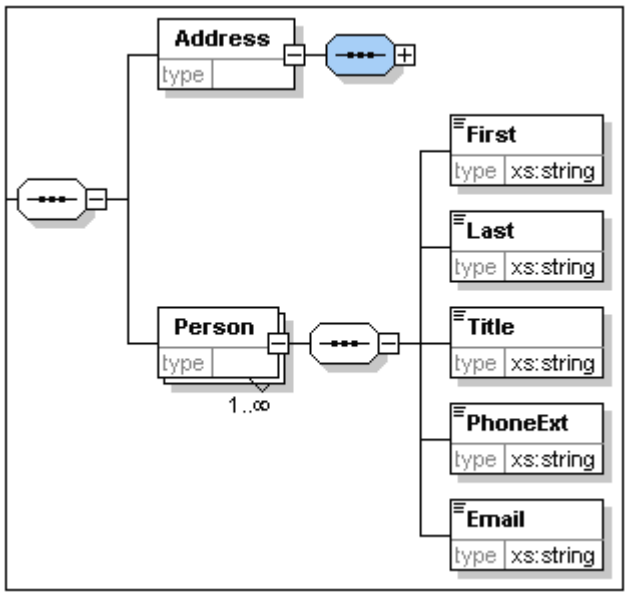
Um das Content Model für Person zu erstellen, gehen Sie folgendermaßen vor:

1. Rechtsklicken Sie auf das Element Person, um das Kontextmenü aufzurufen und wählen Sie Child hinzufügen | Sequence. Der Sequence-Kompositor wird eingefügt.
2. Klicken Sie mit der rechten Maustaste auf den Sequence-Kompositor und wählen Sie Child hinzufügen | Element.
3. Nennen Sie das Element First und betätigen Sie die Tabulatortaste Ihrer Tastatur. Sie gelangen automatisch in das Feld type.

© 2007 Altova GmbH



4. Wählen Sie den Eintrag xs:string aus der Auswahlliste oder geben Sie ihn in das Wertefeld von type ein.
5. Erzeugen Sie weitere vier Elemente mittels der Drag & Drop Methode. Benennen Sie diese der Reihe nach Last, Title, PhoneExt und Email.

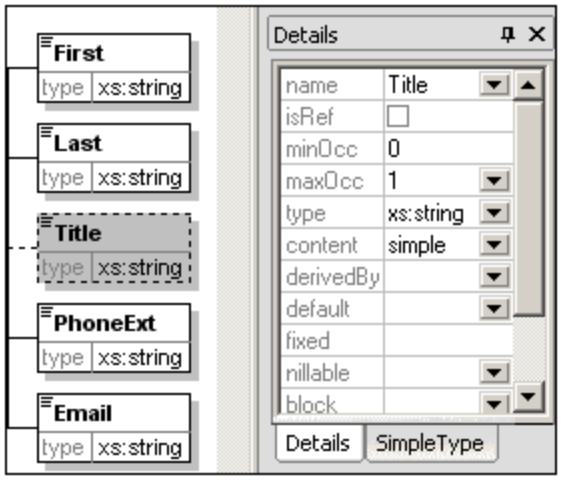


Bitte beachten Sie: Sie können mehrere Elemente auswählen, indem Sie die Strg-Taste gedrückt halten, während Sie die Elemente anklicken. Auf diese Art können Sie mehrere Elemente gleichzeitig kopieren.

Ein Element optional machen

Rechtsklicken Sie auf das Element Title und wählen Sie Optional im Kontextmenü. Die

Umrahmung des Elementes wechselt von durchgezogener Linie auf gestrichelt. Hierdurch wird angezeigt, dass dieses Element optional ist.

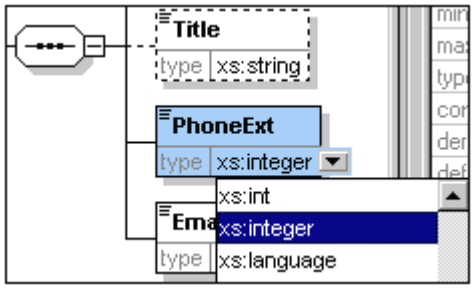


Sie sehen in der Details-Eingabehilfe, dass minOcc=0 und maxOcc=1 ist, d.h. das Element ist optional. Alternativ dazu können Sie das Element auch über das Kontextmenü optional machen und für minOcc=0 einstellen, um das Element zu einem optionalen zu machen.

Den Inhalt eines Elements einschränken

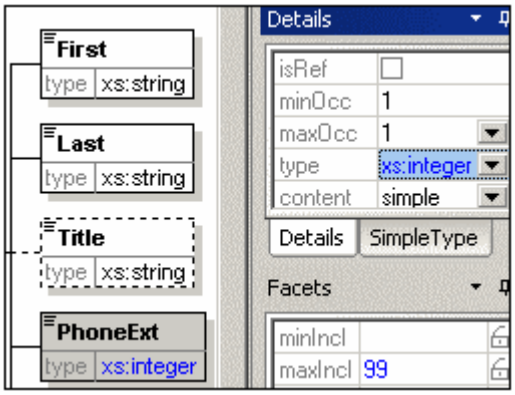
Um das Element PhoneExt als Typ xs:integer mit maximal 2 Stellen zu definieren, gehen Sie folgendermaßen vor:

1. Doppelklicken Sie in das Feld type unter dem Element PhoneExt und wählen Sie den Eintrag xs:integer aus der Auswahlliste oder geben Sie ihn ein.



   Änderungen werden in der Facets-Eingabehilfe angezeigt.
2. Doppelklicken Sie in der Facets-Eingabehilfe in das Feld maxIncl, geben Sie 99 ein und bestätigen Sie dies durch Drücken der Eingabetaste.

© 2007 Altova GmbH



Hierdurch wird definiert, dass alle Durchwahlnummern bis inkl. 99 gültig sind.

3. Wählen Sie die Menüoption Datei | Speichern zum Speichern der Änderungen im Schema.

Bitte beachten Sie:

- Beim Auswählen eines XML-Schema-Datentyps vom Typ simpleType (z.B. xs:string oder xs:date) wird das Content Model automatisch in der Details-Eingabehilfe auf simple (content = simple) gesetzt.
- Beim Hinzufügen eines Kompositors zu einem Element (selection, choice oder all), wird das Inhaltsmodell automatisch auf complex (content = complex) gesetzt.
- Das verwendete Schema ist als `AddressFirst.xsd` im Verzeichnis `XMLSpy2007\Examples\Tutorial` Ihres XMLSpy-Anwendungsordners abgelegt.

2.1.3 Komplexere XML-Schema-Definitionen

Nachdem Sie nun ein einfaches Schema erstellt haben, können wir uns den komplizierteren Aspekten der Schema-Entwicklung widmen.

Ziel
In diesem Abschnitt werden folgende Punkte behandelt:

- Arbeiten mit complexTypes und simpleTypes, die später als die Typen von Schema-Elementen verwendet werden können.
- Erstellen von globalen Elementen und Referenzieren dieser Elemente von anderen Elementen aus
- Erstellen von Attributen und deren Eigenschaften, einschließlich Enumerationswerten.

Wir beginnen diesen Abschnitt mit dem einfachen Schema AddressFirst.xsd, das Sie im ersten Teil dieses Tutorials erstellt haben.

In diesem Abschnitt verwendete Befehle
In diesem Abschnitt des Tutorials wird ausschließlich die Schema/WSDL-Ansicht verwendet. Dabei kommen die folgenden Befehle zum Einsatz:

Diagramm anzeigen (oder Content Model-Ansicht anzeigen). Mit diesem Symbol, das in der Schema-Übersicht links von allen globalen Komponenten angezeigt wird, sehen Sie das Content Model der dazugehörigen globalen Komponente

Globale Elemente anzeigen. Dieses Symbol befindet sich in der linken oberen Ecke des Content Model. Mit diesem Befehl kommen Sie zurück zur Schema-Übersicht, in der alle globalen Komponenten angezeigt werden.

Anhängen. Mit diesem Symbol (in der linken oberen Ecke der Schema-Übersicht) können Sie eine globale Komponente hinzufügen.

Arbeiten mit ComplexTypes und SimpleTypes

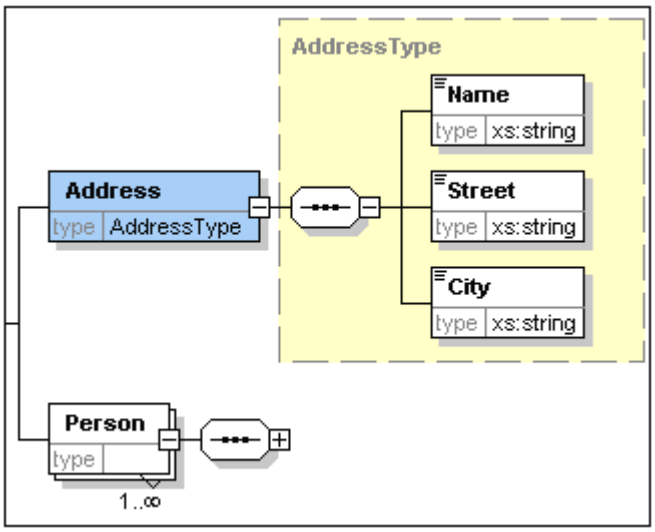
Nachdem Sie ein Element definiert haben, möchten Sie es möglicherweise auch an einer anderen Stelle im Schema verwenden. Dies erreichen Sie dadurch, dass Sie diese Elementdefinition als globalen ComplexType oder als globales Element erstellen. In diesem Abschnitt werden Sie mit globalen ComplexTypes arbeiten. Wir werden zuerst einen ComplexType auf globaler Ebene erstellen und diesen dann erweitern, um ihn in einem Content Model verwenden zu können. Mehr über globale Elemente erfahren Sie weiter hinten in diesem Tutorial.

Erstellen eines globalen complex type

Das Basiselement Address, das wir zuvor definiert haben (und das die Elemente Name, Street und City enthält), kann in verschiedenen Adressformaten wiederverwendet werden. Wir wollen daher diese Elementdefinition als wiederverwendbaren ComplexType erstellen. Gehen Sie dazu folgendermaßen vor:

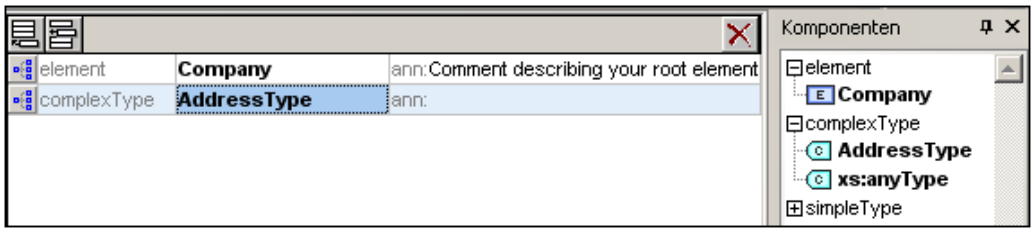
So erstellen Sie einen globales Element vom Typ Complex Type:

1. Klicken Sie in der Content Model-Ansicht mit der rechten Maustaste auf das Element Address.
2. Wählen Sie im Kontextmenü nun die Option Als global definieren | Complex Type. Es wird ein globaler ComplexType mit dem Namen AddressType erstellt, und dem Element Address im Content Model Company wird dieser Typ zugewiesen. Der Inhalt des Elements Address ist das Content Model von AddressType, das in einem gelb hinterlegten Kasten angezeigt wird. Beachten Sie, dass der Datentyp des Elements Address nun AddressType ist.

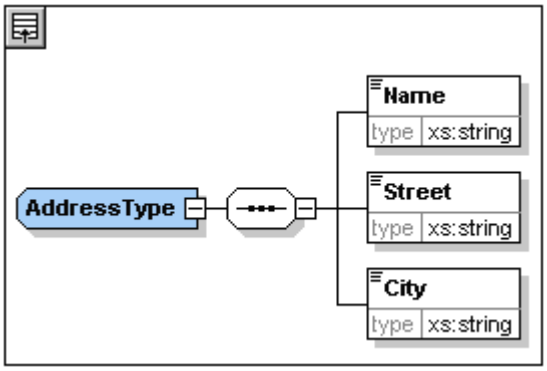


3. Klicken Sie auf das Symbol "Globale Elemente anzeigen" . Damit kehren Sie zur Schema-Übersicht zurück, in der nun alle globalen Komponenten angezeigt werden.
4. Klicken Sie in der Komponenten-Eingabehilfe auf das Erweiterungssymbol für Element- und ComplexType-Einträge, um die jeweiligen Schema-Konstrukte zu sehen.In der Schema-Übersicht werden nun zwei globale Komponenten angezeigt: das Element Company und der ComplexType AddressType. Außerdem wird in der Komponenten-Eingabehilfe der ComplexType AddressType angezeigt.

© 2007 Altova GmbH



5. Klicken Sie auf das Symbol "Content Model-Ansicht" von AddressType, um dessen Content Model anzuzeigen (siehe Abbildung unten). Beachten Sie die Form des ComplexType-Behälters.

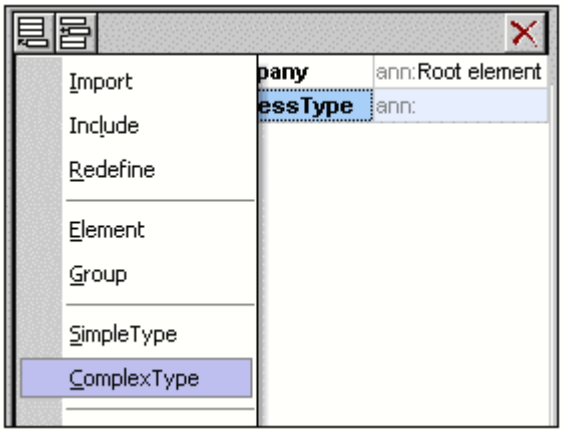


6. Klicken Sie auf das Symbol "Globale Elemente anzeigen", um zur Schema-Übersicht zurückzukehren.

Erweitern einer "complex type" Definition

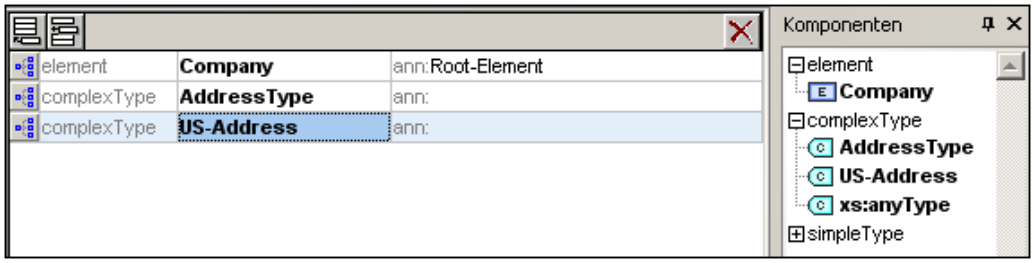
Nun werden wir die globale Komponente AddressType verwenden, um zwei Arten von länderspezifischen Adressen zu erstellen. Hierzu wird zuerst ein neuer complexType erzeugt, welcher auf der Komponente AddressType basiert. Anschließend wird diese Definition erweitert. Gehen Sie dazu folgendermaßen vor:

1. Wechseln Sie zur Schema-Übersicht (Wenn Sie sich in der Content Model-Ansicht befinden, klicken Sie auf "Globale Elemente anzeigen").
2. Klicken Sie auf das Symbol Anhängen (oben links im Komponentenfenster). Daraufhin wird das folgende Menü angezeigt:

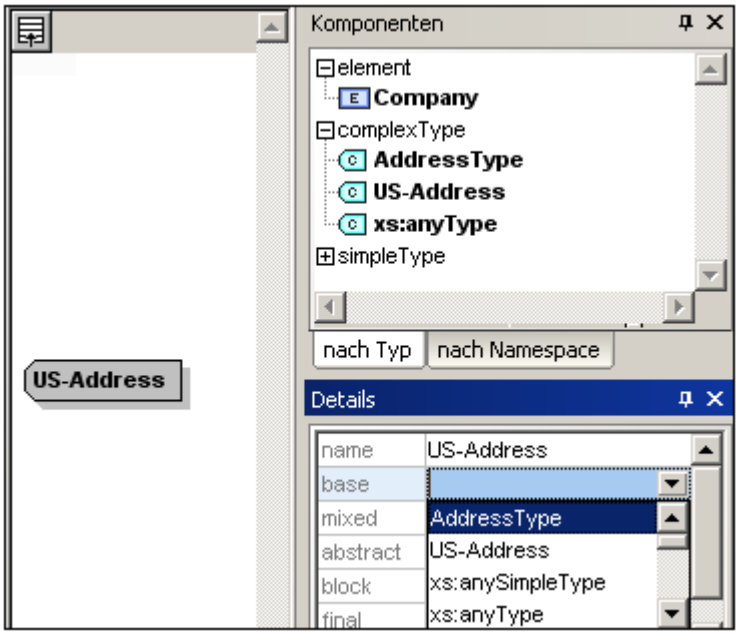


3. Wählen Sie ComplexType aus dem Kontextmenü. Eine neue Leerzeile wird in der Komponentenliste angehängt, in der Sie den Namen der Komponente eingeben können.

4. Geben Sie US-Address ein und bestätigen Sie mit der Eingabetaste. (Sollten Sie den Bindestrich "-" beim Eingeben vergessen, und stattdessen ein Leerzeichen verwenden, wird der Name in Rot angezeigt. D.h., ein nicht erlaubtes Schriftzeichen wurde verwendet.)

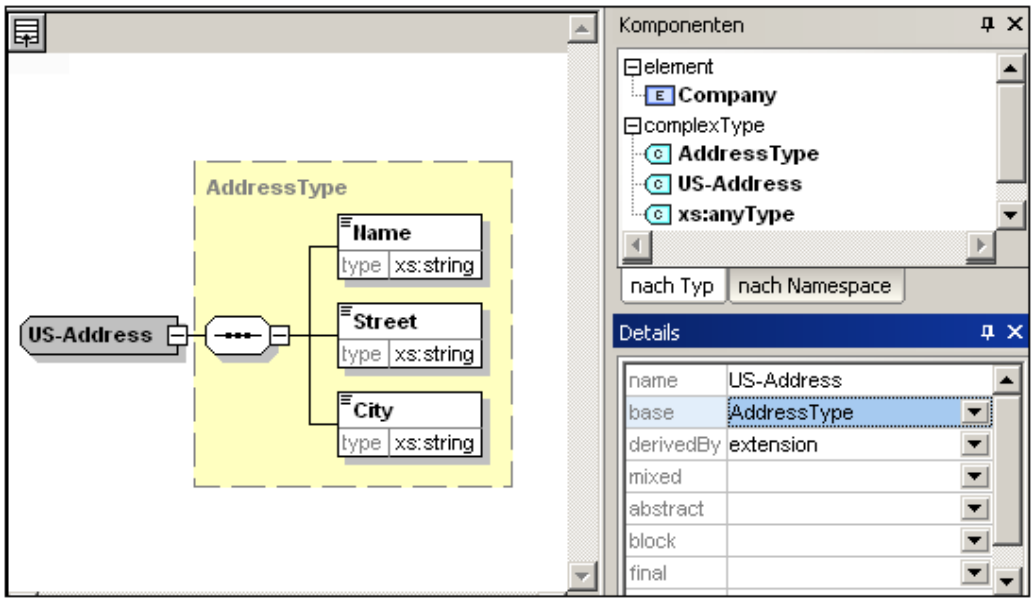


5. Klicken Sie auf das Symbol "Content Model-Ansicht" in der Zeile US-Address, um das Content Model des neuen ComplexType zu sehen. Das Content Model ist leer (siehe Abbildung unten).
6. Klicken Sie in der Details-Eingabehilfe auf das Listenfeld base und wählen Sie AddressType.

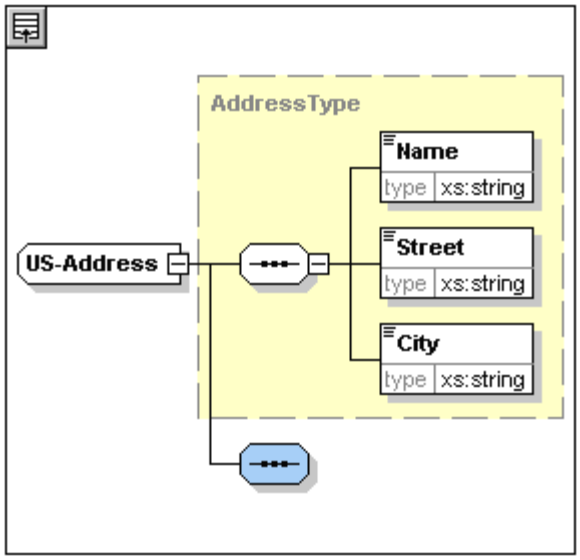


In der Content Model-Ansicht wird nun das Content Model AddressType als Content Model von US-Address (siehe Abbildung unten) angezeigt.

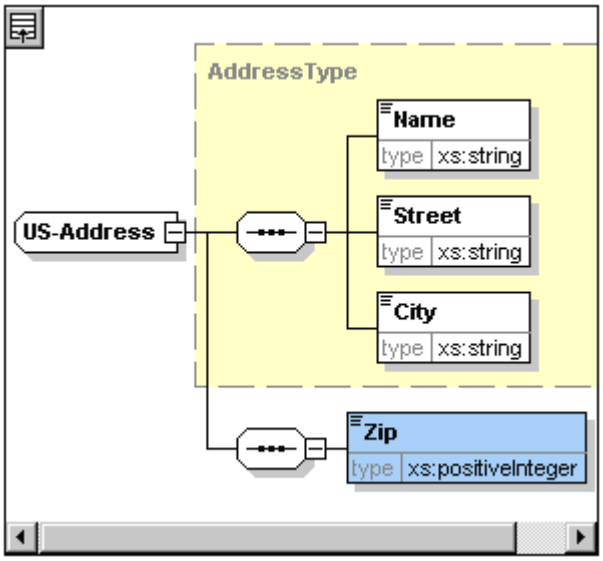
© 2007 Altova GmbH



7. Wir können nun das Content Model des ComplexType US-Address um ein ZIP Code-Element erweitern. Klicken Sie dazu mit der rechten Maustaste auf das Element US-Address und wählen Sie Child hinzufügen | Sequence. Ein neuer Sequence-Kompositor wird außerhalb des Kastens AddressType (Abbildung unten) angezeigt; d.h., es handelt sich um die Erweiterung zu einem Element.



8. Klicken Sie mit der rechten Maustaste auf den neuen Sequence-Kompositor und wählen Sie Child hinzufügen | Element.
9. Benennen Sie das neu erstellte Element mit Zip und drücken Sie die Tabulatortaste. Daraufhin wird der Cursor in das Wertefeld der Typbeschreibungszeile gesetzt.
10. Wählen Sie in der Auswahlliste xs:positiveInteger aus und bestätigen Sie mit der Eingabetaste.



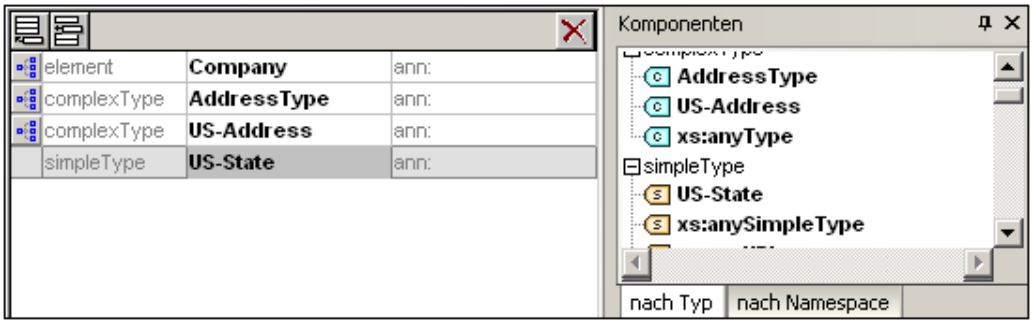
Sie haben nun einen ComplexType namens US-Address, der auf dem ComplexType AddressType basiert und diesen um einen ZIP Code (Postleitzahl) erweitert.

Globale SimpleTypes
Genau wie der ComplexType US-Address auf dem ComplexType AddressType basiert, kann ein Element auch auf einem SimpleType basieren. Der Vorteil ist derselbe wie bei globalen ComplexTypes: Der SimpleType kann wiederverwendet werden. Dazu muss der SimpleType zuerst als globales Element definiert werden. In diesem Tutorial werden wir ein Content Model für US-Bundesstaaten in einem SimpleType definieren. Dieser SimpleType wird als Basis für ein weiteres Element verwendet.

Erstellen eines globalen SimpleType
Um einen globalen SimpleType zu erstellen, müssen Sie einen neuen SimpleType an die Liste der globalen Komponenten anhängen, diesen benennen und seinen Datentyp definieren.

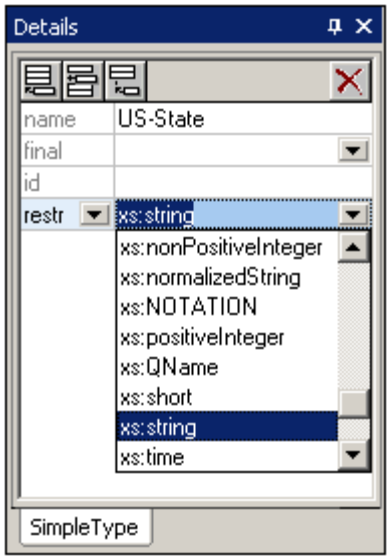
So erstellen Sie einen globalen SimpleType:

1. Wechseln Sie in die Schema-Übersicht (Klicken Sie auf "Globale Elemente anzeigen", wenn Sie sich in der Content Model-Ansicht befinden).
2. Klicken Sie auf das Symbol Anhängen und wählen Sie im Kontextmenü Simple Type.
3. Geben Sie US-State als Elementname ein.
4. Bestätigen Sie mit der Eingabetaste. Der SimpleType US-State wird erstellt und in der Komponenten-Eingabehilfe in die Liste der SimpleTypes aufgenommen (Klicken Sie auf das Erweiterungssymbol des SimpleType-Eintrags, um es zu sehen).



 © 2007 Altova GmbH

5. Platzieren Sie in der Details-Eingabehilfe (Abbildung unten) den Cursor in das Wertefeld von restr und geben Sie xs:string ein oder wählen Sie xs:string in der Dropdown-Liste im Wertefeld restr aus.

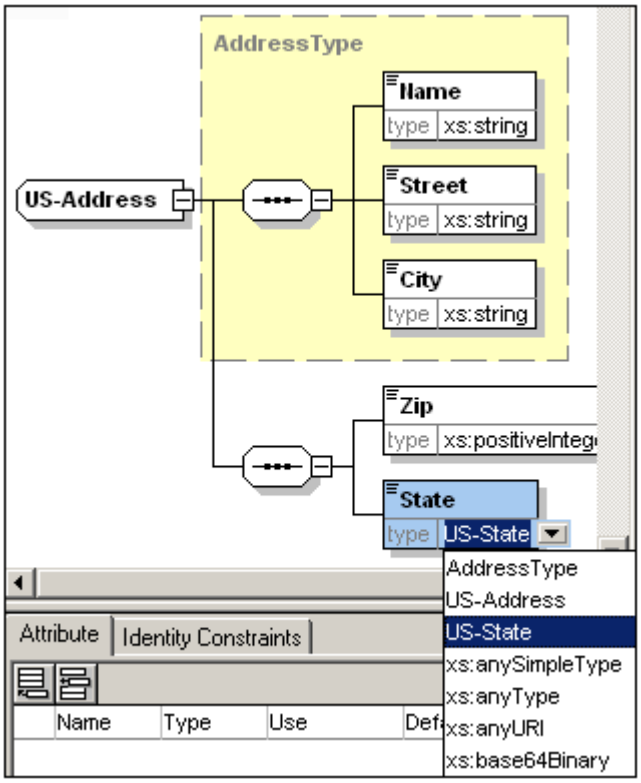


Damit wird ein SimpleType namens US-State vom Datentyp xs:string erstellt. Diese globale Komponente kann nun im Content Model von US-Address verwendet werden.

Verwendung eines globalen SimpleType in einem Content Model

Ein globaler SimpleType kann in einem Content Model dazu verwendet werden, den Typ einer Komponente zu definieren. Wir werden US-State verwenden, um im Content Model von US-Address ein Element namens State zu definieren. Gehen Sie dazu folgendermaßen vor:

1. Klicken Sie in der Schema-Übersicht auf das Symbol "Content Model-Ansicht" von US-Address.
2. Klicken Sie mit der rechten Maustaste auf den unteren Sequence-Kompositor und wählen Sie Child hinzufügen | Element.
3. Geben Sie State als Elementnamen ein.
4. Drücken Sie die Tabulatortaste, um den Cursor in das Wertefeld der Typbeschreibungszeile zu setzen.
5. Wählen Sie in der Dropdown-Liste den Eintrag US-State aus.



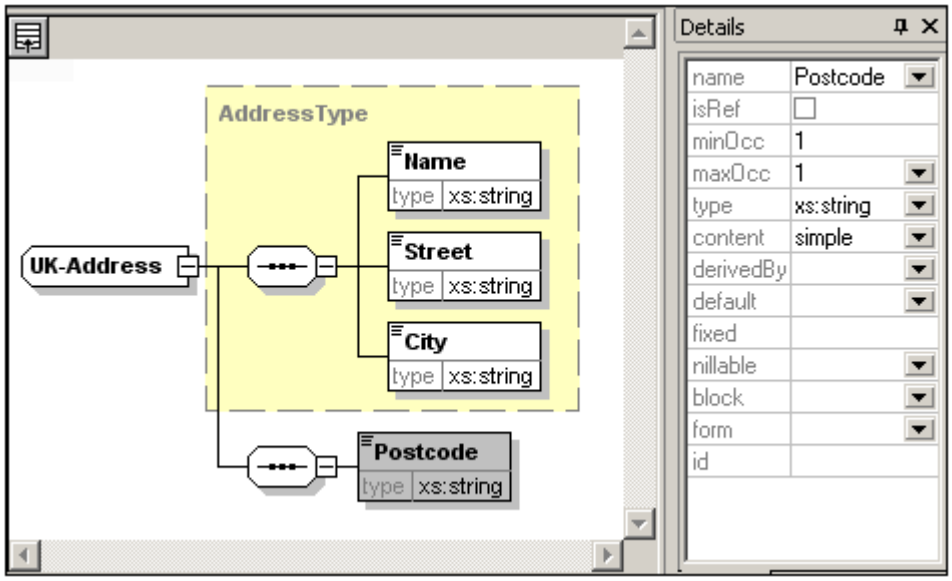
Das Element State basiert nun auf dem SimpleType US-State.

Erstellen eines zweiten ComplexType auf Basis von AddressType
Wir erstellen nun einen globalen ComplexType für Adressen in Großbritannien. Der ComplexType basiert auf dem Element AddressType und wird erweitert, bis er dem englischen Adressformat entspricht.

Gehen Sie folgendermaßen vor:

1. Erstellen Sie in der Schema-Übersicht auf Basis von AddressType einen globalen ComplexType mit dem Namen UK-Address. (base=AddressType).
2. Hängen Sie in der Content Model-Ansicht von UK-Address das neue Element Postcode an und weisen Sie ihm den Typ xs:string zu. bDas Content Model von UK-Address sollte nach erfolgreicher Erstellung folgendermaßen aussehen:

© 2007 Altova GmbH

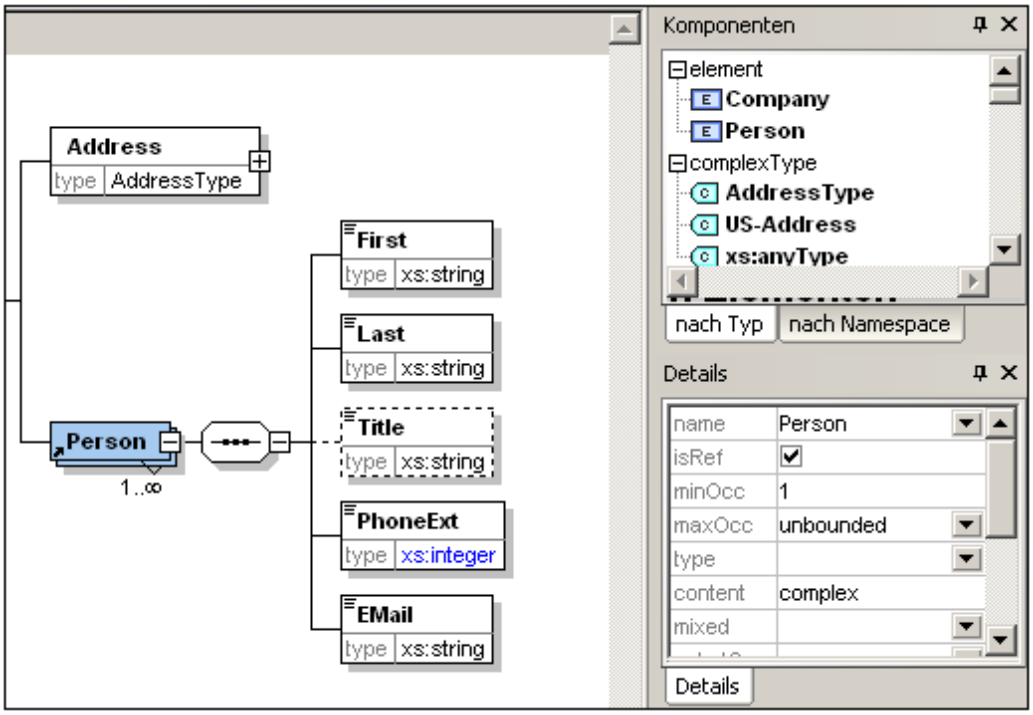


Bitte beachten Sie: In diesem Abschnitt haben Sie globale Simple und ComplexTypes erstellt und anschließend in Content Model-Definitionen verwendet. Der Vorteil von globalen Typen liegt darin, dass diese in mehreren Definitionen wiederverwendet werden können.

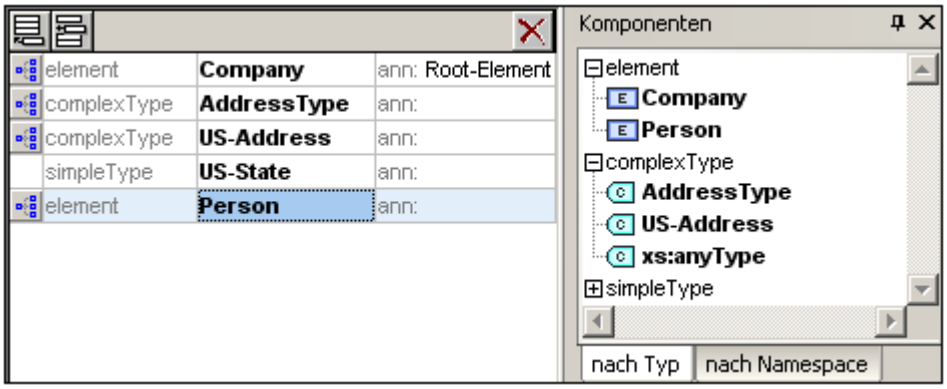
Referenzieren von globalen Elementen

In diesem Abschnitt werden wir das lokal definierte Element Person als globales Element definieren und es dann vom Element Company aus referenzieren.

1. Wechseln Sie in die Schema-Übersicht (Globale Elemente anzeigen).
2. Klicken Sie auf das Symbol "Diagramm anzeigen" in der Zeile Company.
3. Klicken Sie mit der rechten Maustaste auf das Element Person und wählen Sie Als global definieren | Element. Ein kleines "link"-Symbol wird in das Element Person eingefügt, das anzeigt, dass das Element das als global deklarierte Element Person referenziert. Das Kontrollkästchen isRef in der Details Eingabehilfe ist nun auf aktiv gesetzt.

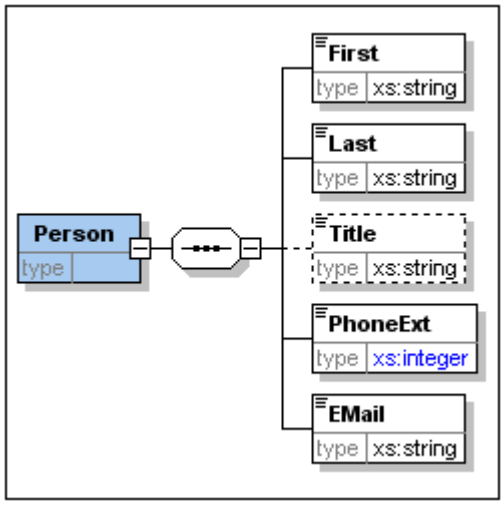


4. Klicken Sie auf das Symbol "Globale Elemente anzeigen" um zur Schema-Übersicht zurückzukehren. Das Element Person wird nun als globales Element aufgelistet und erscheint auch in der Komponenten-Eingabehilfe.



5. Klicken Sie in der Komponenten-Eingabehilfe auf das Element Person, um das Content Model des globalen Elements Person anzuzeigen.

© 2007 Altova GmbH



Bitte beachten Sie, dass das Kästchen für das globale Element kein Link-Symbol hat, da es sich hierbei um das referenzierte Element und nicht das referenzierende Element handelt. Nur das referenzierende Element ist mit einem Link-Symbol gekennzeichnet.

Bitte beachten Sie:

- Ein Element, das ein globales Element referenziert, muss denselben Namen haben, wie das globale Element, das es referenziert.
- Globale Deklarationen geben keine Auskunft über die Positionierung der Elemente innerhalb des XML-Dokumentes. Sie bestimmen nur den Inhalt des jeweiligen Elementes. Nur wenn eine globale Deklaration von einer anderen Komponente aus referenziert wird, wird ihre Position im XML-Dokument definiert.

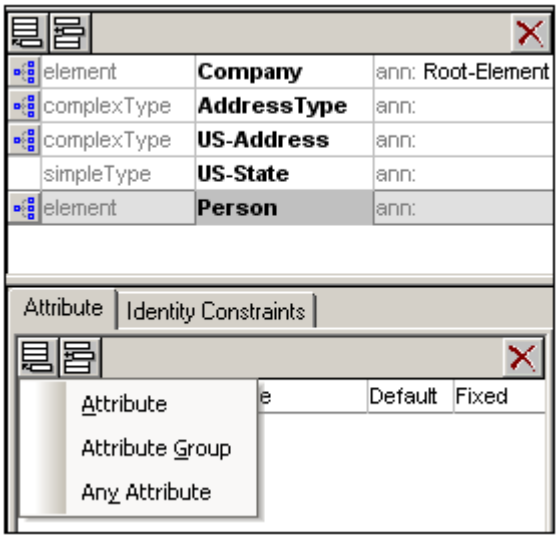
Ein global deklariertes Element kann an mehreren Stellen wiederverwendet werden. Es unterscheidet sich insofern von einem global deklarierten ComplexType, als sein Content Model nicht geändert werden kann, ohne dass auch das globale Element selbst geändert wird. Wenn Sie das Content Model eines Elements ändern, das ein globales Element referenziert, so wird auch das Content Model des globalen Elements geändert und damit auch das Content Model aller anderen Elemente, die das globale Element referenzieren.

Attribute und Attribut-Enumerationen

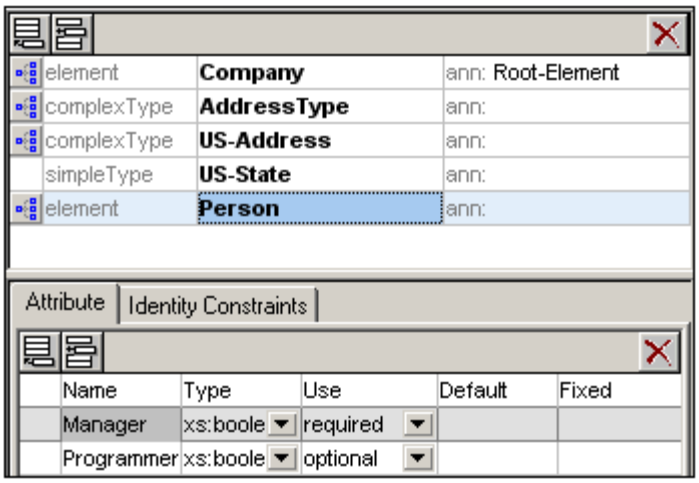
In diesem Abschnitt lernen Sie, wie man Attribute und Enumerationen für Attribute erstellt.

Definieren von Element-Attributen

1. Klicken Sie in der Schema-Übersicht auf das Element `Person` um es aktiv zu setzen.
2. Klicken Sie auf das Symbol "Anhängen", (oben links im Fenster "Attribute/Identity Constraints", das den unteren Bereich des Schema-Übersichtsfensters einnimmt) und wählen Sie den Eintrag "Attribute".



3. Geben Sie Manager in das Namensfeld ein.
4. Wählen Sie in der Dropdown-Liste Type den Eintrag `xs:boolean`.
5. Wählen Sie in der Dropdown-Liste Use den Eintrag required.



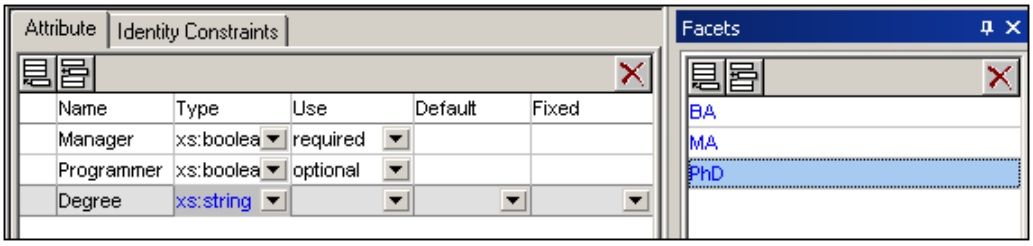
6. Erstellen Sie nun auf dieselbe Art ein Attribut Programmer mit einem Type=`xs:boolean` und Use=`optional`.

Definieren von Enumerationen für Attribute
Enumerationen sind Werte, die bei einem bestimmten Attribut zulässig sind. Wenn der Wert des Attributs im XML-Instanzdokument nicht in einer im XML-Schema definierten Enumeration enthalten ist, ist das Dokument ungültig. Wir werden Enumerationen für das Attribut Degree des Elements `Person` erstellen.

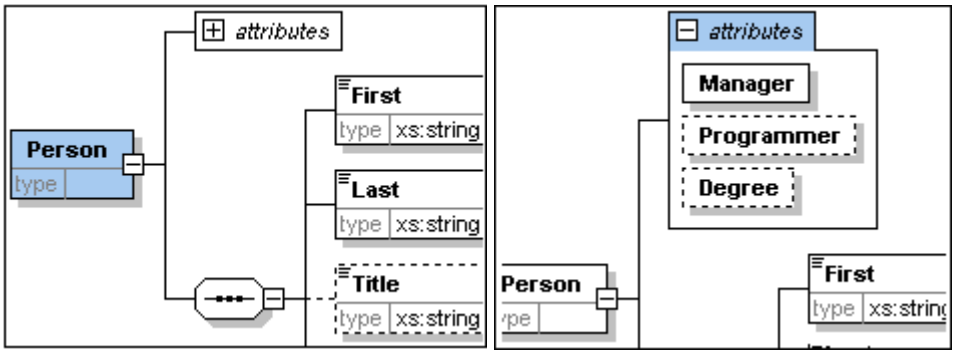
Gehen Sie folgendermaßen vor:

1. Klicken Sie in der Schemaübersicht auf das Element `Person`, um es zum aktiven zu machen.
2. Klicken Sie auf das Anhängen-Symbol oben links im Attribut-Fenster und wählen Sie den Eintrag "Attribute".
3. Geben Sie Degree als Attributsnamen ein und wählen Sie `xs:string` als Attribut-Type.
4. Klicken Sie in der Facets-Eingabehilfe auf das Register Enumerations (siehe Abbildung unten).

© 2007 Altova GmbH



5. Klicken Sie auf dem Register "Enumerations" auf das Anhängen-Symbol .
6. Geben Sie `BA` (Bachelor of Arts) ein und bestätigen Sie mit der Eingabetaste.
7. Mit derselben Methode geben Sie zwei weitere Werte in die Enumerations-Liste ein: `MA` und `PhD`.
8. Klicken Sie auf das Symbol "Content Model Ansicht" des Elements `Person`.



Die zuvor definierten Attribute werden in der Content Model-Ansicht angezeigt. Wenn Sie auf das Erweiterungssymbol klicken, werden alle für dieses Element definierten Attribute angezeigt. Sie können diesen Anzeigemodus ein- und ausschalten, indem Sie die Menüoption **Schema-Design | Ansicht** konfigurieren wählen und auf die Kontrollkästchen für Attribute und Identity Constraints klicken.

9. Klicken Sie auf das Symbol "Globale Elemente anzeigen", um zur Schema-Übersicht zurückzukehren.

Speichern des fertigen Schemas
Bitte beachten Sie: Bevor Sie Ihre Schema-Datei speichern, sollten Sie zuerst die vorhandene Datei `AddressLast.xsd`, die mit XMLSpy mitgeliefert wird, umbenennen (z.B. in `AddressLast_original.xsd`), um sie nicht zu überschreiben.

Speichern Sie das fertige Schema unter einem Namen Ihrer Wahl (**Datei | Speichern unter**). Wir empfehlen Ihnen, es unter dem Namen `AddressLast.xsd` zu speichern, da die XML-Datei, die Sie im nächsten Teil des Tutorials erstellen werden, auf dem Schema `AddressLast.xsd` basieren wird.

2.1.4 Schemanavigation und Dokumentation

Nach Fertigstellung Ihres XML-Schemas sollten Sie sich nun mit einigen Navigationskürzeln vertraut machen und etwas über die Schemadokumentation, die Sie in XMLSpy generieren können, erfahren. Diese Themen werden in den folgenden Unterabschnitten erörtert.

In diesem Abschnitt verwendete Befehle
Wir verwenden in diesem Abschnitt des Tutorials ausschließlich die Schema/WSDL-Ansicht.

Dabei werden die folgenden Befehle verwendet:

Diagramm anzeigen (oder Content Model-Ansicht anzeigen). Mit diesem Symbol, das in der Schema-Übersicht links von allen globalen Komponenten angezeigt wird, sehen Sie das Content Model der dazugehörigen globalen Komponente.

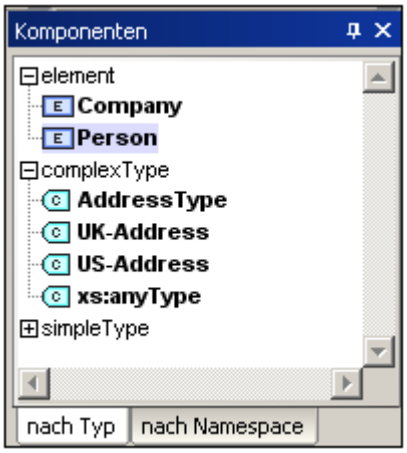
Schemanavigation

In diesem Abschnitt wird beschrieben, wie Sie effizient innerhalb der Schema/WSDL-Ansicht navigieren können. Probieren Sie diese Navigationsverfahren am besten aus, um sich damit vertraut zu machen.

Anzeigen des Content Model einer globalen Komponente

Globale Komponenten, die ein Content Model haben können, sind ComplexTypes, Elemente und Elementgruppen. Die Content Model-Ansicht dieser Komponenten lässt sich auf folgende Arten aufrufen:

- Klicken Sie in der Schema-Übersicht auf das Symbol "Diagramm anzeigen" links vom Namen der Komponente.
- Doppelklicken Sie entweder in der Schema-Übersicht oder in der Content Model-Ansicht in der Komponenten-Eingabehilfe auf das Element, den ComplexType oder die Elementgruppe (*siehe Abbildung unten*).



Wenn Sie in der Komponenten-Eingabehilfe auf einer der anderen globalen Komponenten (SimpleType, Attribut, Attributgruppe) doppelklicken, erscheint diese Komponente in der Schema-Übersicht (da diese Komponenten kein Content Model haben) markiert.

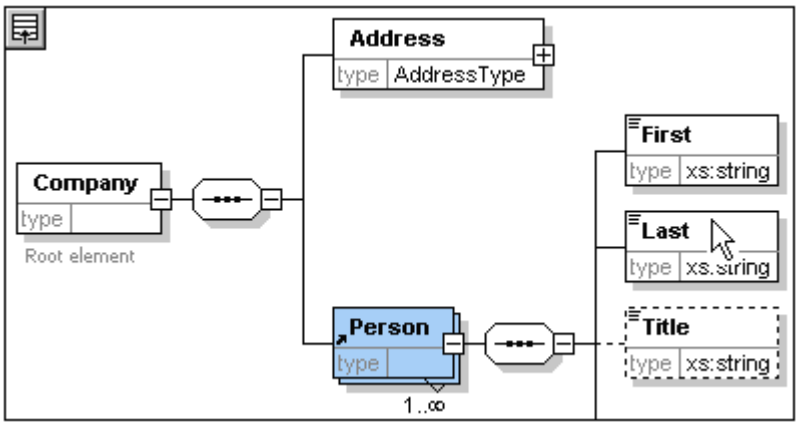
Das Anzeigen der Komponenten durch Doppelklick funktioniert in der Komponenten-Eingabehilfe sowohl auf dem Register "By Type" als auch auf dem Register "By Namespace".

So gelangen Sie von einem referenzierenden Element aus zur Definition eines globalen Elements:
Wenn ein Content Model ein Element enthält, das ein globales Element referenziert, können Sie das Content Model dieses globalen Elements oder das seiner darin enthaltenen Komponenten direkt aufrufen, indem Sie die Strg-Taste gedrückt halten, während Sie auf das gewünschte Element doppelklicken.

Wenn Sie z.B. während der Ansicht des Content Model von `Company` die **Strg**-Taste gedrückt

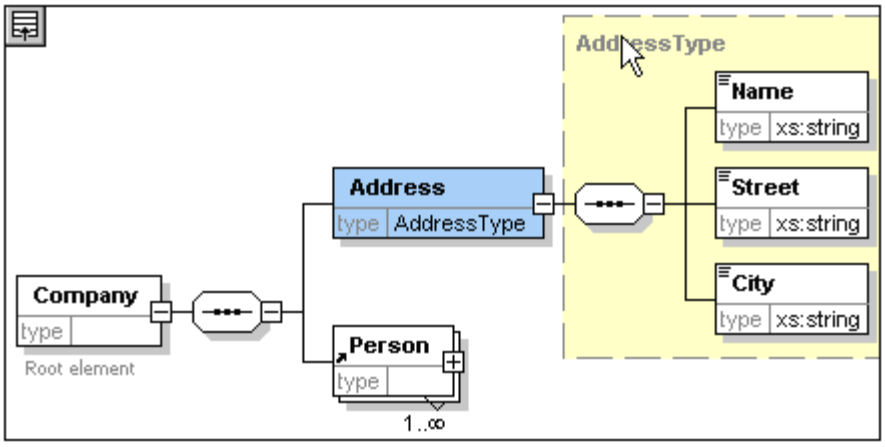
© 2007 Altova GmbH

halten, während Sie auf `Last` doppelklicken, wird das Content Model `Person` geöffnet und das Element `Last` darin wird markiert.

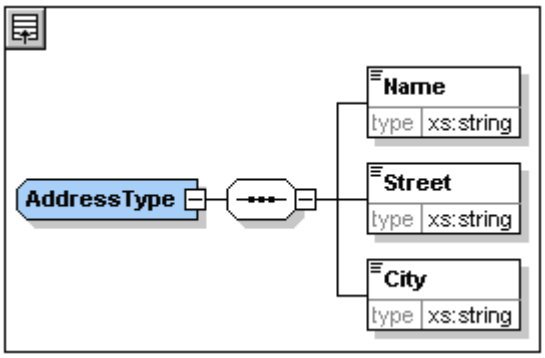


Wenn das Element `Last` markiert ist, werden alle seine Eigenschaften sofort in den entsprechenden Eingabehilfen und Informationsfenstern angezeigt.

So gelangen Sie zur Definition eines ComplexType
ComplexTypes werden oft als Elementtyp in einem Content Model verwendet. Um die Definition eines ComplexType direkt von einem Content Model aus aufzurufen, doppelklicken Sie im gelben Kästchen auf den Namen des ComplexType (siehe Mauszeiger in der Abbildung unten).



Auf diese Art gelangen Sie zur Content Model-Ansicht des ComplexType.



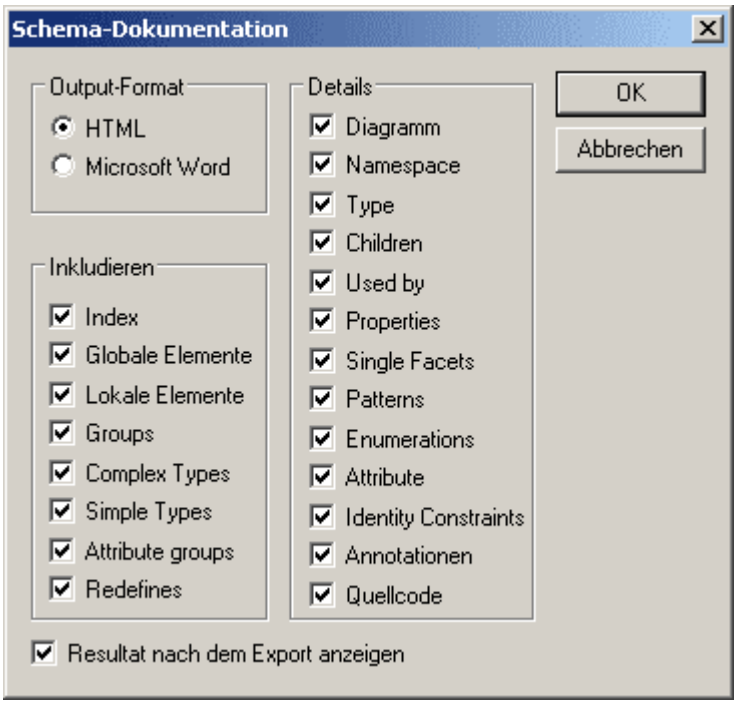
Bitte beachten Sie: Wie auch bei referenzierten globalen Elementen können Sie direkt zu einem Element innerhalb der ComplexType-Definition gehen, indem Sie die **Strg**-Taste gedrückt halten und auf das gewünschte Element im Content Model, das den ComplexType enthält, doppelklicken.

Schema-Dokumentation

XMLSpy verfügt über Funktionen zur Erstellung einer detaillierten Dokumentation von XML-Schemas im HTML- und MS Word-Format. Sie können die Komponenten und die Genauigkeit der Dokumentation auswählen. Miteinander in Zusammenhang stehende Komponenten werden im HTML- und im Word-Dokument durch Hyperlinks verbunden. Um Dokumentation im MS Word-Format zu generieren, muss MS Word auf Ihrem Computer (oder Netzwerk) installiert sein.

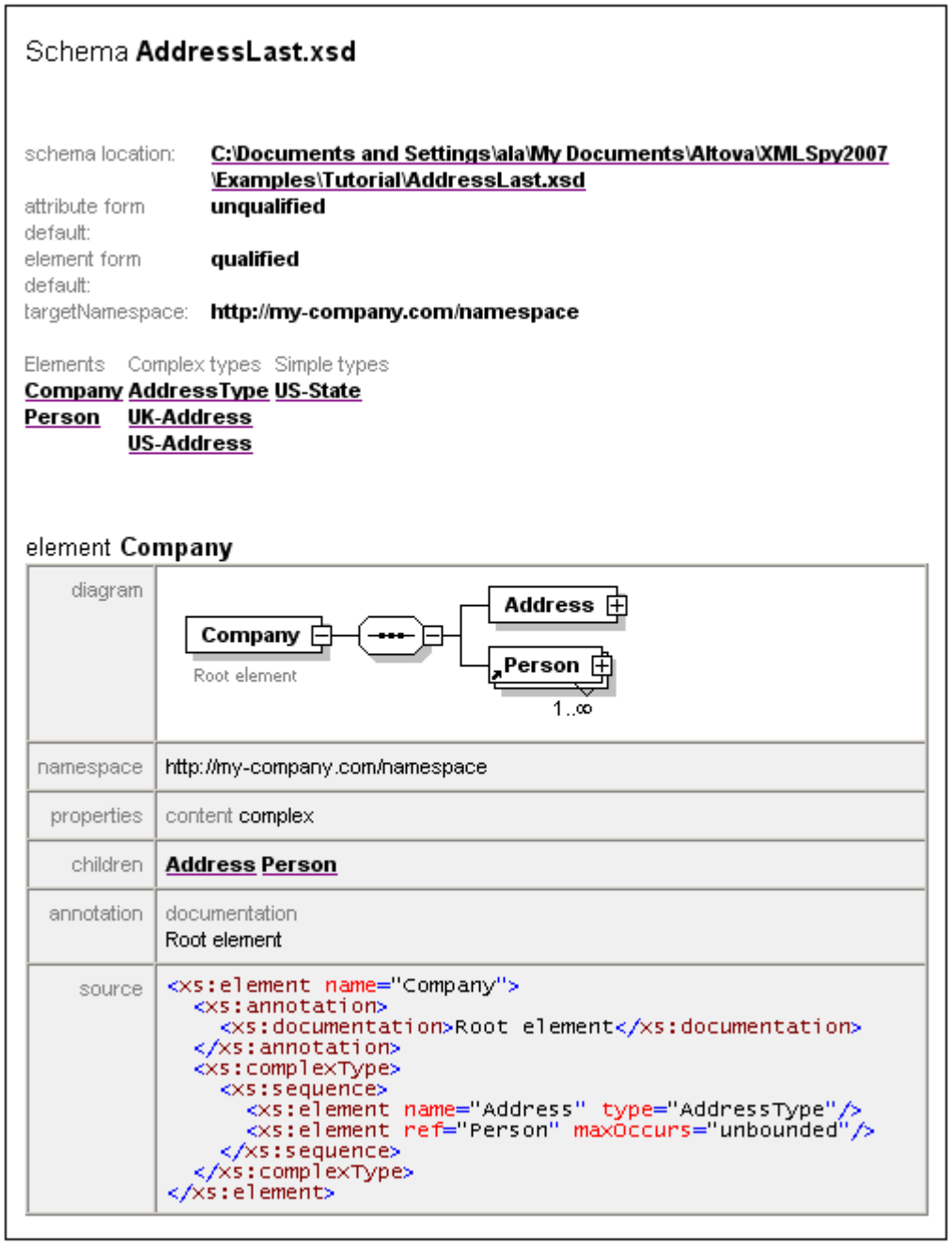
Sie werden in diesem Abschnitt Dokumentation für das XML-Schema `AddressLast.xsd` generieren. Gehen Sie dazu folgendermaßen vor:

1. Wählen Sie die Menüoption **Schema-Design | Schema-Dokumentation generieren,** um das Dialogfeld "Schema-Dokumentation" aufzurufen.



2. Wählen Sie das Ausgabeformat HTML und bestätigen Sie mit OK.
3. Wählen Sie im Dialogfeld "Speichern unter" das Verzeichnis und geben Sie einen Namen für die HTML-Datei ein (z.B. `AddressLast.html`). Bestätigen Sie mit **Speichern**. Das HTML-Dokument wird in der Browser-Ansicht von XMLSpy angezeigt. Wenn Sie auf einen Link klicken, gelangen Sie zur damit verknüpften Komponente.

© 2007 Altova GmbH



Schema **AddressLast.xsd**

schema location: **C:\Documents and Settings\ala\My Documents\Altova\XMLSpy2007\Examples\Tutorial\AddressLast.xsd**
attribute form default: **unqualified**
element form default: **qualified**
targetNamespace: **http://my-company.com/namespace**

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

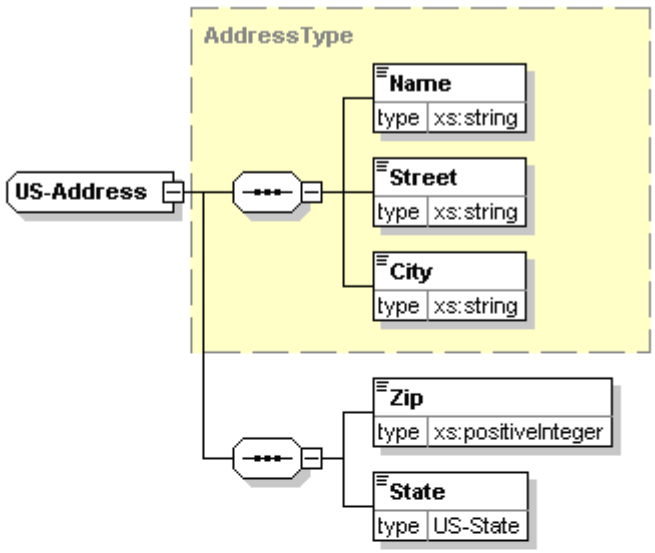
element **Company**

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>`

Das oben abgebildete Diagramm stellt die **erste Seite** der Schema-Dokumentation im HTML-Format dar. Falls Komponenten anderer Schemas miteinbezogen wurden, werden diese ebenfalls dokumentiert.

complexType **US-Address**

diagram	
namespace	http://my-company.com/namespace
type	extension of **AddressType**
properties	name US-Address base AddressType
children	**Name Street City Zip State**
source	<xs:complexType name="US-Address"> <xs:complexContent> <xs:extension base="AddressType"> <xs:sequence> <xs:element name="Zip" type="xs:positiveInteger"/> <xs:element name="State" type="US-State"/> </xs:sequence> </xs:extension> </xs:complexContent> </xs:complexType>

Das oben abgebildete Diagramm stellt dar, wie ComplexTypes dokumentiert werden.

© 2007 Altova GmbH

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>

Das oben abgebildete Diagramm zeigt an, wie Elemente und SimpleTypes dokumentiert werden.

Sie können nun die MS Word-Ausgabeoption ausprobieren. Das Word-Dokument wird in MS Word geöffnet. Um Hyperlinks im MS Word-Dokument zu verwenden, halten Sie die **Strg**-Taste gedrückt, während Sie auf den Link klicken.

2.1.5 Erstellen von XML Dokumenten

In diesem Abschnitt lernen Sie:
Erstellen eines neuen XML Dokumentes in XMLSpy, Verwenden von verschiedenen Ansichten und Eingabehilfen für erleichtertes Bearbeiten und Validieren von Daten in XMLSpy.

Ziel

In diesem Abschnitt werden folgende Punkte behandelt:

- Erstellen eines neuen XML-Dokumentes basierend auf dem Schema `AddressLast.xsd`.
- Definition des Elementtyps, um für das Element bei der Validierung ein erweitertes Content Model für dieses Element bereitzustellen.
- Einfügen von Elementen und Attributen und Eingabe von Inhalt für diese in der Grid- und der Text-Ansicht mit Hilfe der intelligenten Eingabehilfen.
- Kopieren von XML-Daten von XMLSpy in ein MS Excel-Blatt, Einfügen von Daten in MS Excel und Zurückkopieren der Daten nach XMLSpy. Diese Funktion steht in der Datenbank/Tabellen-Ansicht der Grid-Ansicht zur Verfügung.
- Sortieren von XML-Elementen mit Hilfe der Sortierfunktion der Datenbank/Tabellen-Ansicht.
- Validieren des XML-Dokumentes.
- Ändern des Schemas, so dass dreistellige Telefondurchwahlnummern eingegeben werden können.

In diesem Abschnitt verwendete Befehle

Datei | Neu erstellt eine neue XML-Datei.

Ansicht | Text wechselt in die Text-Ansicht.

Ansicht | Enhanced Grid wechselt in die Enhanced Grid-Ansicht.

XML | Tabelle | Als Tabelle anzeigen zeigt Mehrfachinstanzen eines einzelnen Elementtyps auf einer einzelnen hierarchischen Ebene als Tabelle an. Diese Ansicht des Elements nennt sich Datenbank/Tabellen-Ansicht (oder einfach Tabellenansicht). Das Symbol dient zum Wechseln zwischen der Tabellen,- und der normalen Enhanced Grid-Ansicht.

F7 Zum Kontrollieren, ob das Dokument wohlgeformt ist.

F8 Validiert das XML-Dokument gegen das damit verknüpfte Schema bzw. die DTD.

Öffnet die dazugehörige DTD oder XML-Schema-Datei.

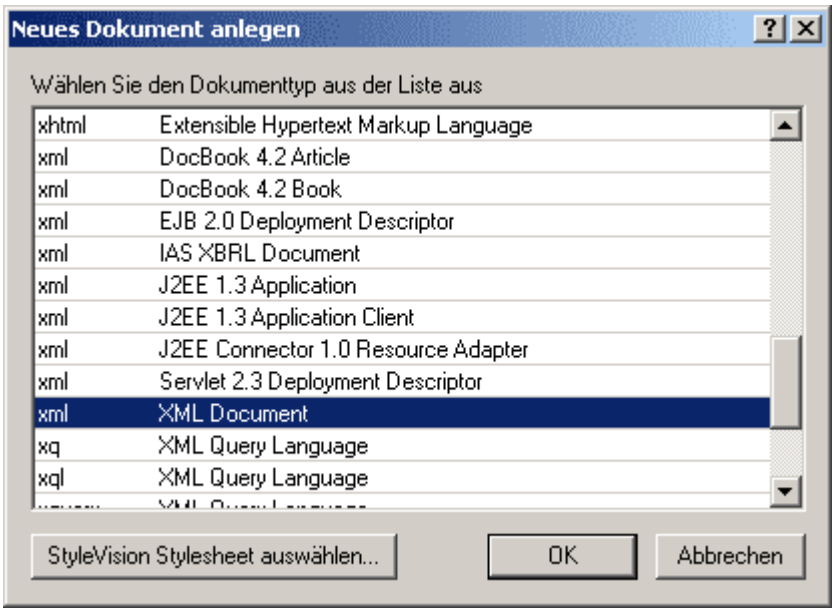
Erstellen eines neuen XML-Dokuments

Beim Erstellen einer neuen XML-Datei in XMLSpy haben Sie die Option, dieser ein Schema (DTD oder XML-Schema) zugrunde zu legen oder auch nicht. In diesem Abschnitt werden wir eine neue Datei erstellen, die auf dem zuvor im Rahmen dieses Tutorials erzeugten Schema `AddressLast.xsd` basiert.

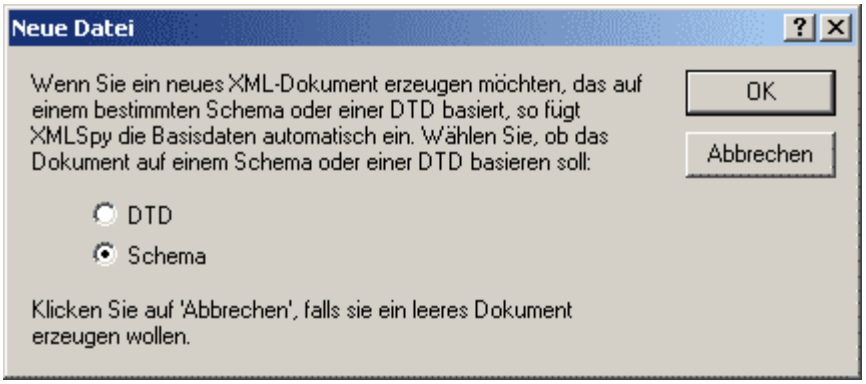
So erstellen Sie eine neue XML-Datei:

1. Wählen Sie die Menüoption **Datei | Neu**, um Dialogfeld "Neues Dokument anlegen" aufzurufen.

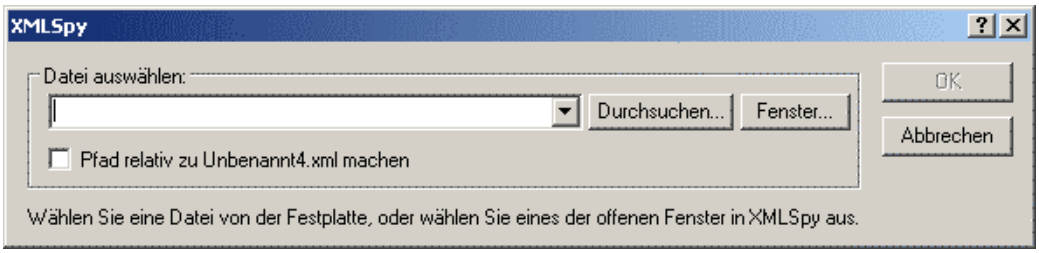
© 2007 Altova GmbH



2. Wählen Sie den Eintrag . `xml XML Document` im Dialogfeld und bestätigen Sie mit OK. Im nächsten Fenster werden Sie gefragt, ob das XML-Dokument auf einer DTD oder einem Schema basieren soll.

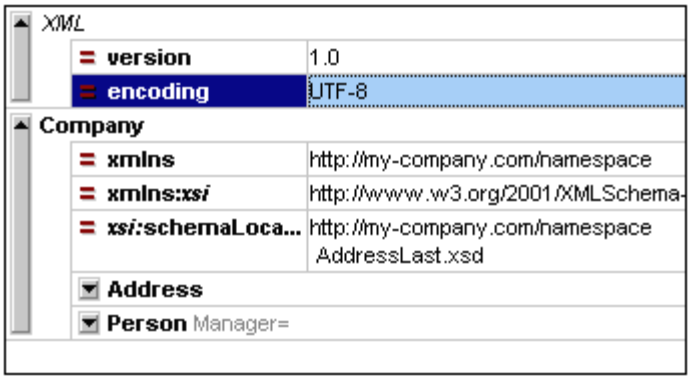


3. Wählen Sie das Optionsfeld "Schema" und bestätigen Sie mit OK. Sie werden nun aufgefordert, die Schema-Datei auszuwählen, auf der Ihr XML-Dokument basieren soll.

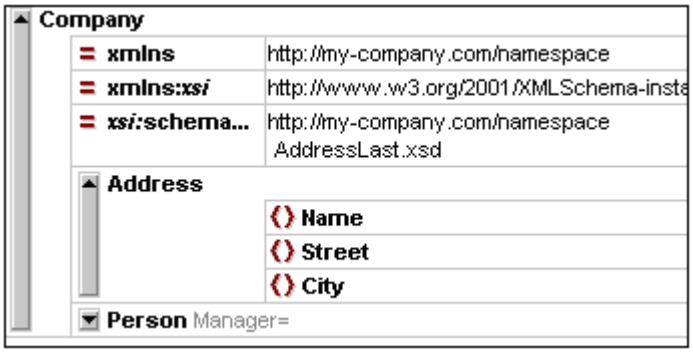


4. Wählen Sie "Durchsuchen" oder "Fenster", um die Schema-Datei zu suchen. Bei Klicken auf die Schaltfläche "Fenster" werden alle Dateien, die in XMLSpy und Projekten geöffnet sind, aufgelistet. Wählen Sie `AddressLast.xsd` und bestätigen Sie Ihre Auswahl mit OK. Im Hauptfenster wird ein neues XML-Dokument geöffnet. Das Dokument enthält alle Hauptelemente, die im Schema definiert wurden.
5. Klicken Sie auf das Register **Grid**, um die Enhanced Grid-Ansicht auszuwählen.
6. In der Grid-Ansicht ist das gesamte Dokument ausgewählt. Wenn Sie auf ein Element klicken, wird die Auswahl auf dieses Element reduziert. Ihr Dokument sollte etwa

folgendermaßen aussehen:

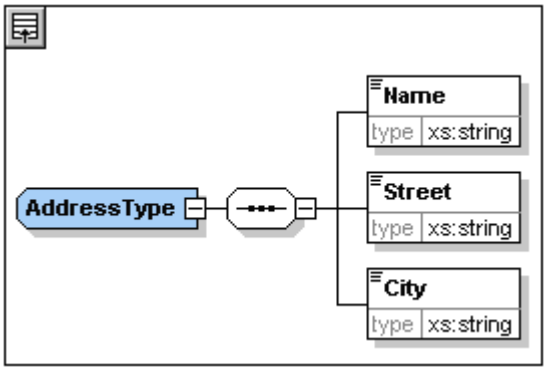


7. Klicken Sie auf das Symbol neben Address, um die Child-Elemente von Address anzuzeigen. Ihr Dokument sollte aussehen wie dieses:



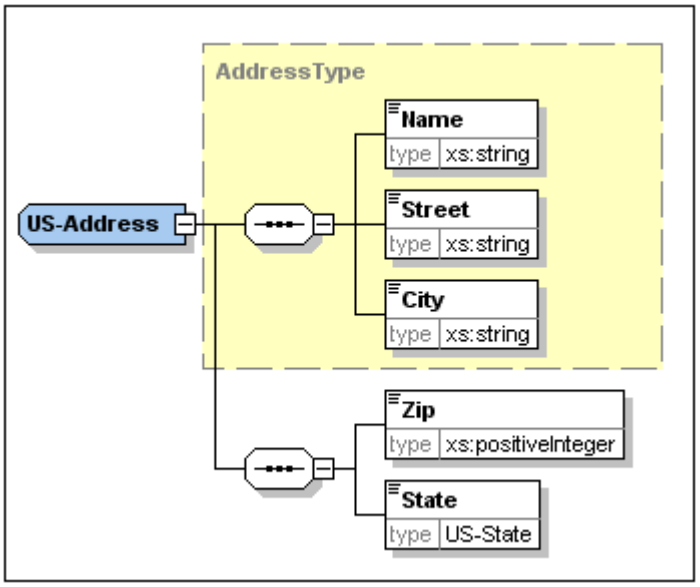
Festlegen des Typs eines Elements

Die in der Grid-Ansicht angezeigten Child-Elemente von `Address` sind die, welche für den globalen ComplexType `AddressType` (*Content Model dazu in der Abbildung unten*) definiert wurden.



Wir möchten jedoch einen spezifischen US- oder UK-Adresstyp verwenden anstelle des generischen Adresstyps. Sie werden sich erinnern, dass wir im Schema `AddressLast.xsd` globale ComplexTypes für `US-Address` und `UK-Address` erstellt haben, indem wir den ComplexType `AddressType` erweitert haben. Unten sehen Sie das Content Model von `US-Address`.

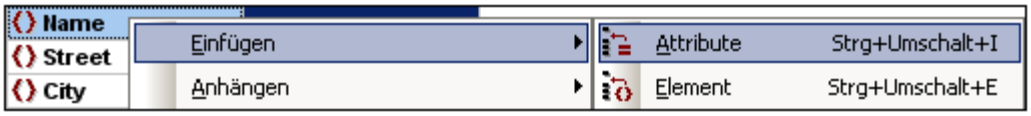
© 2007 Altova GmbH



Um festzulegen, dass das Element `Address` im XML-Dokument einem der erweiterten `Address`-Typen entsprechen muss (`US-Address` oder `UK-Address`) und nicht dem generischen `AddressType`, müssen wir den erforderlichen erweiterten ComplexType als ein Attribut des Elements `Address` definieren.

Gehen Sie folgendermaßen vor:

1. Klicken Sie in der Grid-Ansicht mit der rechten Maustaste auf das Element `Name` und wählen Sie im Kontextmenü **Einfügen | Attribute**.



   Dem Element `Address` wird ein Attributfeld hinzugefügt.
2. Geben Sie `xsi:type` als Name des Attributs ein.
3. Drücken Sie die Tabulatortaste um in das nächste (Werte)-Feld zu gelangen. Es erscheint ein Popup-Menü (*siehe unten*), in dem eine Liste der verfügbaren ComplexTypes angezeigt wird.

4. Wählen Sie `US-Address` aus der Auswahlliste und bestätigen Sie mit der Eingabetaste.

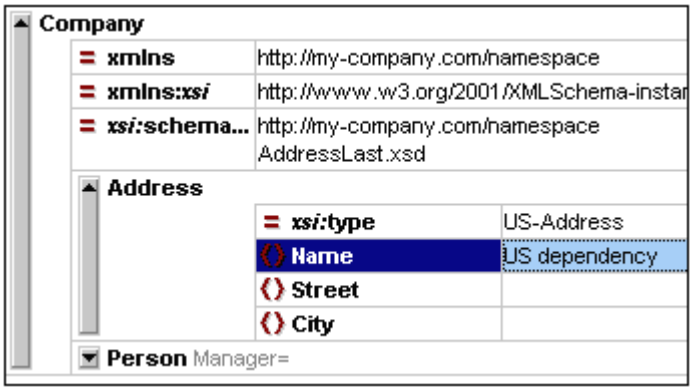
Bitte beachten Sie: Das vorgesetzte `xsi` ermöglicht die Verwendung von speziellen XML-Schema-spezifischen Befehlen im XML-Dokument. Sie haben soeben einen Typ für das Element `Address` festgelegt. Nähere Informationen dazu finden Sie in der XML Schema-Spezifikation.

Eingabe von Daten in der Grid-Ansicht

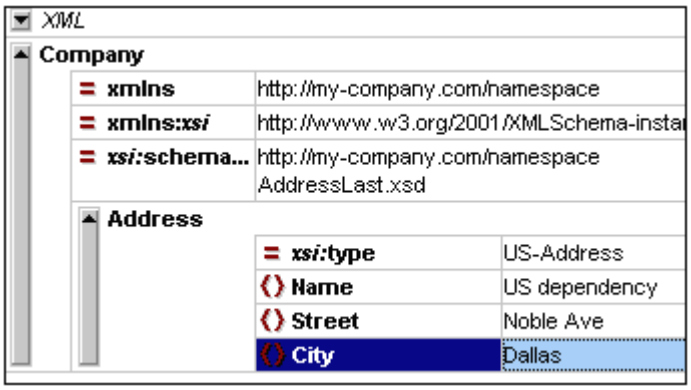
Sie können jetzt Daten in Ihr XML-Dokument eingeben.

Gehen Sie folgendermaßen vor:

1. Doppelklicken Sie in das Wertefeld `Name` (oder verwenden Sie die Pfeiltasten) und geben Sie `US dependency` ein. Bestätigen Sie mit der **Eingabetaste**.



2. Geben Sie auf dieselbe Art einen `Street` und `City` Namen ein (z.B. `Noble Ave.` und `Dallas`).
3. Klicken Sie auf das Element `Person` und drücken Sie die **Entf**-Taste, um es zu löschen. (Wir werden es im nächsten Abschnitt des Tutorials wieder hinzufügen). Daraufhin erscheint nun das gesamte Element `Address` markiert.
4. Klicken Sie auf ein beliebiges Child-Element des Elements `Address`, um die Auswahl für alle Child-Elemente von `Address` mit Ausnahme des gewählten Elements aufzuheben. Ihr XML-Dokument sollte folgendermaßen aussehen:



Eingabe von Daten in der Textansicht

Die Textansicht eignet sich aufgrund ihrer intelligenten Editierfunktionen für DTD/XML-Schema ideal zum Editieren der eigentlichen Daten und des Markup-Codes von XML-Dateien.

Funktionen zum Bearbeiten der Struktur

© 2007 Altova GmbH

Die Textansicht bietet eine Reihe von Funktionen zur Bearbeitung der Struktur, die das Editieren größerer Textabschnitte vereinfachen. Dazu zählen die folgenden Funktionen, die Sie durch Anklicken des entsprechenden Symbols aktivieren bzw. deaktivieren können.

Aktiviert/deaktiviert die Zeilennummerierung

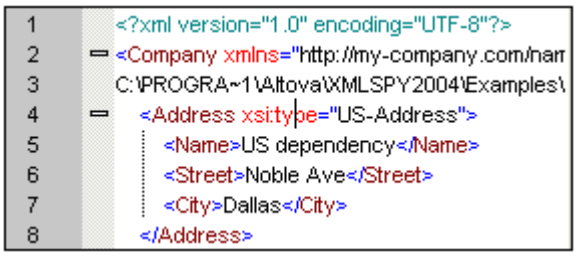
Aktiviert/deaktiviert die Klappleiste

Aktiviert/deaktiviert die Lesezeichenfunktion

Fügt Lesezeichen ein und entfernt diese

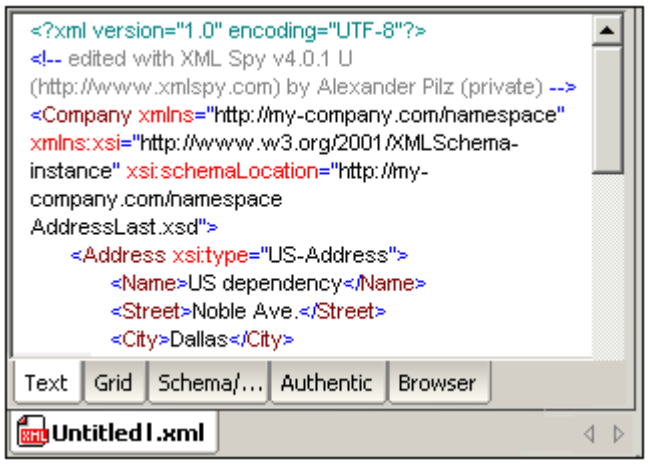
Aktiviert/deaktiviert die Einrücklinien

In der unten gezeigten Abbildung sehen Sie die Textansicht der XML-Datei, die sich in Bearbeitung befindet. Alle Sonderfunktionen der Textansicht sind aktiviert. Aus Gründen der Übersichtlichkeit werden diese Funktionen (Zeilennummerierung, Einrücklinien usw.) in weiterer Folge in dieser Anleitung in Textansichten nicht mehr angezeigt. Nähere Informationen zur Textansicht finden Sie im Benutzerhandbuch.



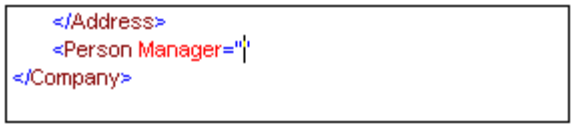
Bearbeitung in der Textansicht
In diesem Abschnitt werden wir Daten in der Textansicht eingeben und bearbeiten, um die Funktionen der Textansicht zu erläutern. Gehen Sie folgendermaßen vor:

1. Wählen Sie die Menüoption **Ansicht | Text** oder klicken Sie auf das Register Text.Sie sehen nun das XML-Dokument in seiner Textform mit der Syntaxfärbung.

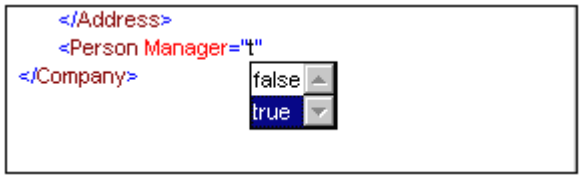


2. Setzen Sie den Cursor hinter den End-Tag des Elements "Address" und drücken Sie die Eingabetaste um eine neue Zeile einzufügen.
3. Geben Sie ein "Kleiner als..." Zeichen (<) an dieser Stelle ein. Eine Auswahlliste aller an dieser Stelle (laut Schema) zulässigen Elemente erscheint. Da an dieser Stelle nur das Element `Person` zulässig ist, wird nur dieses eine Element in der Liste angezeigt.

4. Wählen Sie den Eintrag Person aus. Das Element `Person` wird mit seinem Attribut `Manager` eingefügt, wobei sich der Cursor innerhalb des Wertefelds des Attributs `Manager` befindet.



5. Wählen Sie in der Dropdown-Liste des Attributs `Manager` den Eintrag `true` aus.



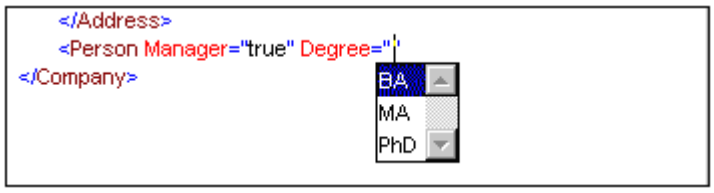
Nach Betätigen der Eingabetaste wird der Wert `true` an der Cursorposition eingefügt.

6. Setzen Sie den Cursor an das Ende der Zeile (oder betätigen Sie die "Ende"-Taste Ihrer Tastatur). Drücken Sie die Leerzeichentaste einmal. Daraufhin wird eine Dropdown-Liste geöffnet, die diesmal eine Liste von an dieser Stelle zulässigen

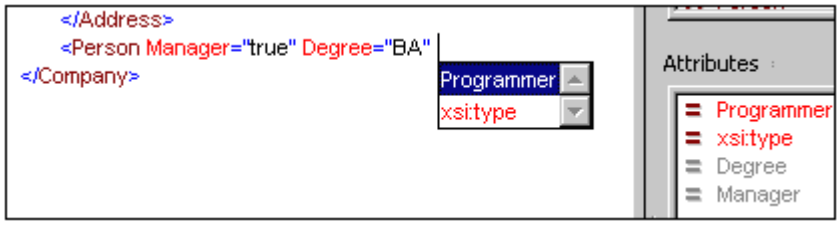
 © 2007 Altova GmbH

Attributen enthält. In der Attribut-Eingabehilfe sind verfügbare Attribute rot markiert. Das Attribut "Manager" ist ausgegraut, da es bereits verwendet wurde.

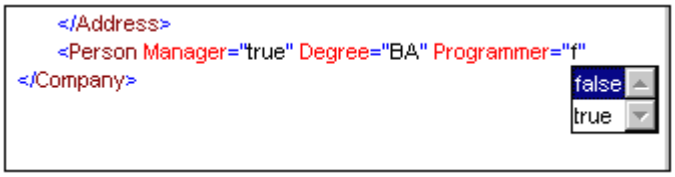
7. Wählen Sie `Degree` durch Betätigen des 'Nach unten'-Pfeiles und drücken Sie die Eingabetaste. Eine weitere Liste wird geöffnet. Hier können Sie eine der vordefinierten Enumerationen auswählen (`BA`, `MA` oder `PhD`).



8. Wählen Sie `BA` aus der Liste, drücken Sie die Eingabetaste und setzen Sie den Cursor durch Drücken der Ende-Taste an das Ende der Zeile. Drücken Sie die Leerzeichentaste. Nun sind `Manager` und `Degree` in der Attribut-Eingabehilfe ausgegraut.



9. Wählen Sie `Programmer` aus der Liste und drücken Sie die Eingabetaste.

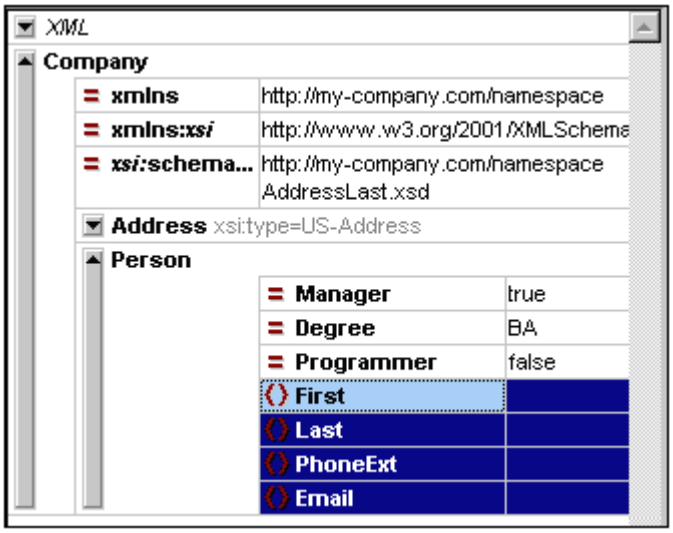


10. Geben Sie den Buchstaben "`f`" ein und drücken Sie die Eingabetaste.
11. Setzen Sie den Cursor ans Ende der Zeile, und geben Sie das "größer als..." Zeichen (>) ein. XMLSpy fügt automatisch alle erforderlichen Child-Elemente von `Person` ein. (Beachten Sie, dass das optionale Element `Title` nicht eingefügt wird). Jedes Element hat einen Start- und einen End-Tag aber keinen Inhalt.

```
<?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</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>
```

Sie könnten nun die Personendaten hier eingeben, doch wollen wir nun in die Grid-Ansicht wechseln, damit Sie sehen, wie einfach sich Ansichten beim Bearbeiten eines Dokuments wechseln lassen.

Wechseln zur Grid-Ansicht
Um zur Grid-Ansicht zu wechseln, wählen Sie die Menüoption **Ansicht | Enhanced Grid** oder klicken Sie auf das Register Grid. Die neu hinzugefügten Child-Elemente von `Person` erscheinen markiert.



Wir wollen das Dokument nun validieren und etwaige Fehler korrigieren.

Validierung des Dokuments

XMLSpy bittet zwei Funktionen zur Überprüfung des XML-Dokuments:

- Wohlgeformtheitprüfung
- Validierung

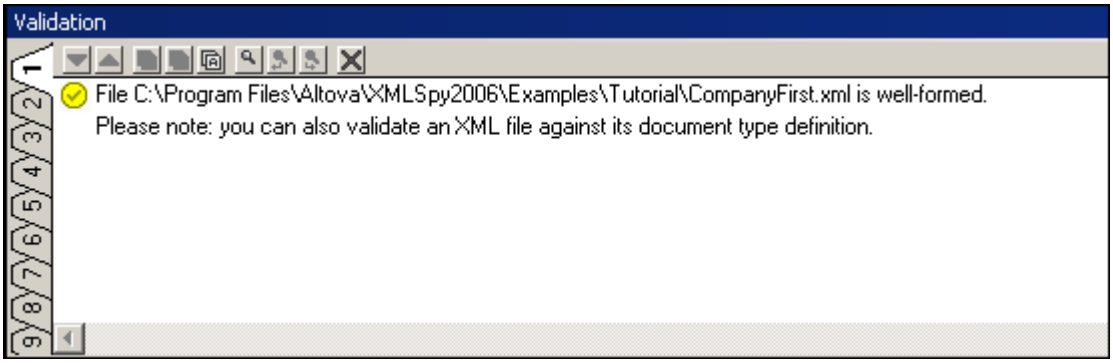
Wenn ein Fehler bei einer dieser beiden Überprüfungen gefunden wird, muss das Dokument entsprechend geändert werden.

© 2007 Altova GmbH

Überprüfung der Wohlgeformtheit
Ein XML-Dokument ist wohlgeformt, wenn es zu jedem Start-Tag einen End-Tag gibt, Elemente korrekt verschachtelt sind und keine Zeichen fehlen oder an der falschen Stelle stehen (z.B. eine Entity ohne das Semikolon usw.).

Wählen Sie zum Prüfen der Wohlgeformtheit die Menüoption **XML | Wohlgeformtheit prüfen** oder drücken Sie die Taste **F7** oder klicken Sie auf . Am unteren Rand des Hauptfensters erhalten Sie eine Meldung, welche anzeigt, ob das Dokument wohlgeformt ist.

Beachten Sie, dass das Validierungsfenster neun Ausgaberegister enthält. Das Ergebnis der Validierung wird immer auf dem aktuellen Register angezeigt. Sie können daher die Wohlgeformtheit für eine Schemadatei auf Register 1 überprüfen und das Ergebnis aufbewahren, indem Sie zu Register 2 wechseln, bevor Sie das nächste Schema-Dokument validieren (andernfalls wird Register 1 durch das Validierungsergebnis überschrieben.)

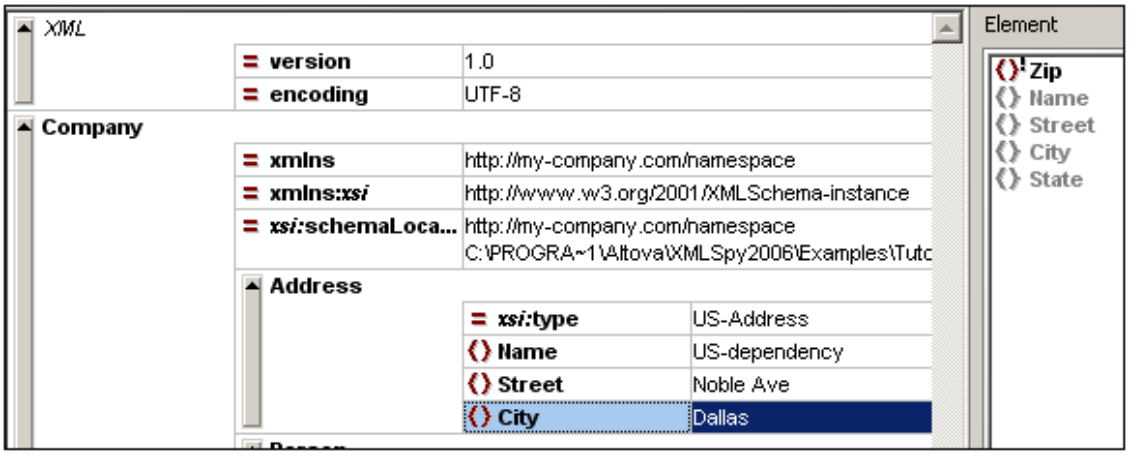


Bitte beachten Sie: Bei der Wohlgeformtheitsprüfung nicht geprüft wird, ob die Struktur der XML-Datei schemakonform ist. Dies wird bei der Validierung der Datei überprüft.

Überprüfung der Gültigkeit (Validierung)
Ein XML-Dokument ist gemäß einem Schema gültig, wenn es von Struktur und Inhalt mit den Schema-Spezifikationen übereinstimmt.

Wählen Sie die Menüoption **XML | Validieren** oder drücken Sie die Taste **F8** oder klicken Sie auf . Im unteren Bereich des Hauptfensters wird eine Fehlermeldung angezeigt, dass die Datei nicht gültig ist. Nach dem Element `City` in `Address` werden obligatorische Elemente erwartet. Wenn Sie Ihr Schema überprüfen, werden Sie feststellen, dass der ComplexType `US-Address` (dem aufgrund von `xsi:type` attribute dieses `Address`-Element entsprechen muss) ein Content Model hat, in dem auf das Element `City` ein Element `Zip` und ein Element `State` folgen muss.

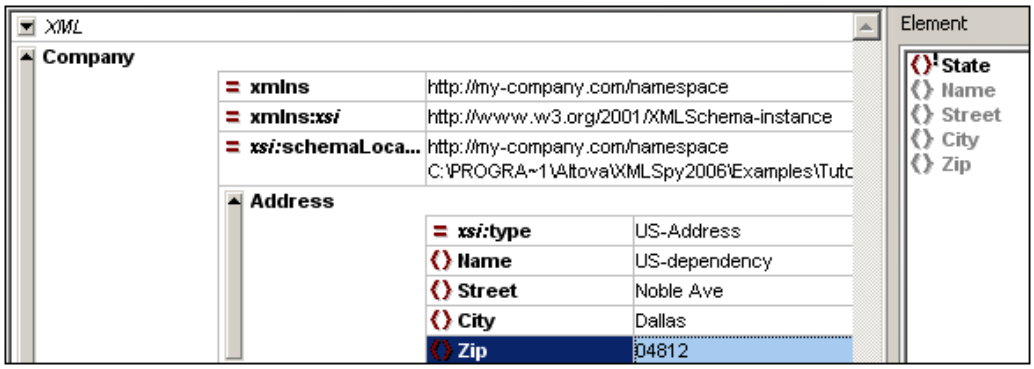
Beheben von Validierungsfehlern
Das Element, welches den Fehler verursacht hat, wird angezeigt.



Bitte beachten Sie hierzu auch die Eingabehilfe (oben rechts), bei welcher ein Ausrufungszeichen "!" vor dem Element `Zip` anzeigt, dass es sich hierbei um ein **obligatorisches** Element handelt.

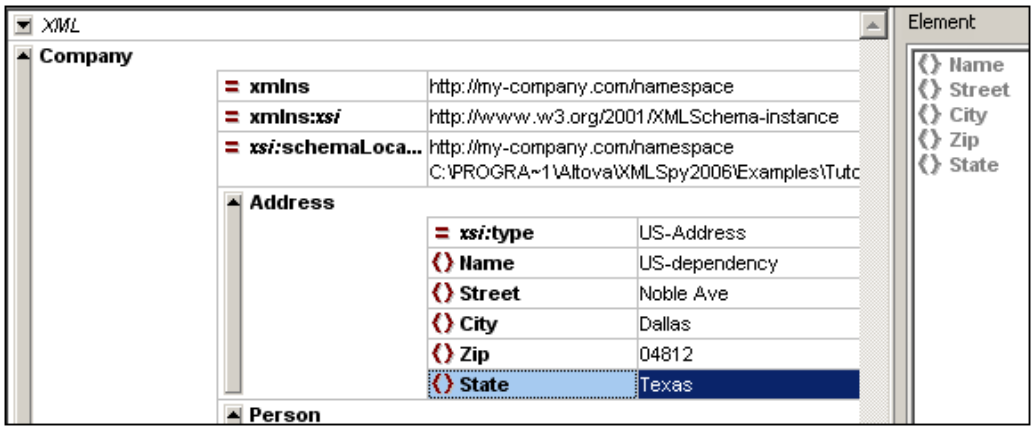
Um den Validierungsfehler zu beheben, gehen Sie folgendermaßen vor:

1. Doppelklicken Sie auf das Element `Zip` in der Element-Eingabehilfe. Hierdurch wird das Element `Zip` unter dem Element `City` eingefügt. (Das Register "Anhängen" ist in der Element-Eingabehilfe aktiv.)
2. Drücken Sie die Tabulatortaste, geben Sie eine Postleitzahl (Zip) ein - (z.B. `04812`) und bestätigen Sie mit der Eingabetaste. Die Element-Eingabehilfe zeigt nun an, dass das Element `State` obligatorisch ist (ein Ausrufezeichen ist dem Element vorangestellt). *Siehe Screenshot unten*.



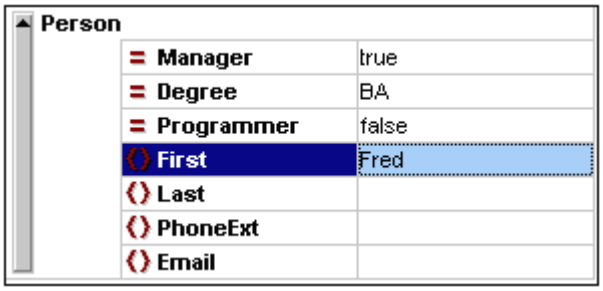
3. Doppelklicken Sie in der Element-Eingabehilfe auf das Element `State`, drücken Sie die Tabulatortaste und geben Sie den Namen des Bundesstaates ein - z.B. `Texas`. Bestätigen Sie mit der Eingabetaste. Die Element-Eingabehilfe enthält nun nur noch ausgegraute Elemente. Hierdurch wird angezeigt, dass alle obligatorischen Child-Elemente von `Address` eingefügt worden sind.

© 2007 Altova GmbH

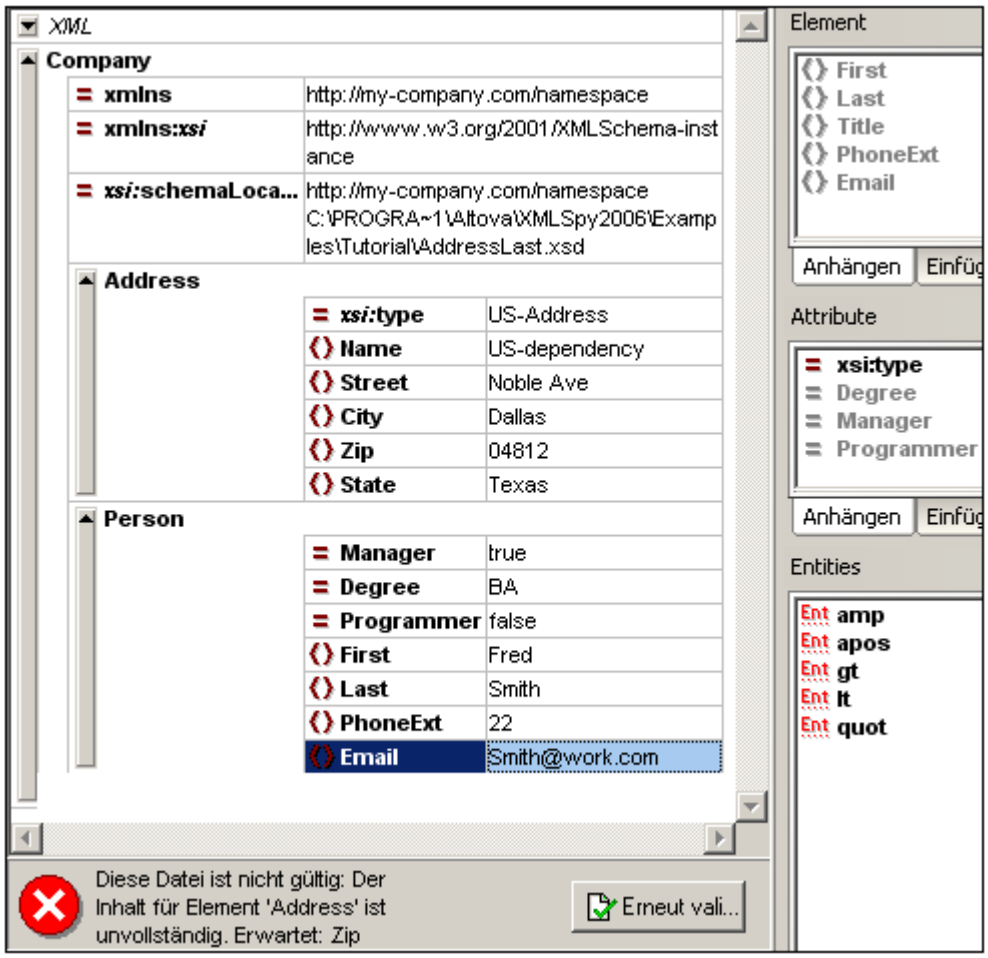


Fertigstellen des Dokuments und erneutes Validieren
Stellen Sie nun das Dokument fertig (geben Sie die Daten für das Element `Person` ein), bevor Sie es erneut validieren. Gehen Sie folgendermaßen vor:

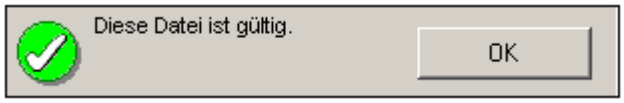
1. Klicken Sie auf das Wertefeld des Elements `First` und geben Sie den Vornamen der Person ein - z.B. `Fred`, bestätigen Sie mit der Eingabetaste.



2. Geben Sie nun auf dieselbe Art Daten für alle Child-Elemente, also `Last`, `PhoneExt` und `Email` ein. Beachten Sie, dass der Wert von `PhoneExt` eine Ganzzahl mit einem Maximalwert von 99 sein muss (da dies der gültige Wertebereich ist, den Sie im Schema definiert haben). Ihr XML-Dokument sollte nun so aussehen:



3. Klicken Sie auf die Schaltfläche **Revalidate,** um das Dokument neuerlich zu prüfen. Diesmal erhalten Sie die Nachricht "`Diese Datei ist gültig`".



4. Das XML Dokument ist nun gegen das Schema validiert und korrekt. Klicken Sie auf OK zum Bestätigen und Schließen der Nachricht.
5. Wählen Sie die Menüoption **Datei | Speichern** und benennen Sie das Dokument - z.B. `CompanyFirst.xml`. Beachten Sie, dass es im Ordner Tutorial bereits eine fertige Datei mit dem Namen `CompanyFirst.xml` gibt, d.h. Sie sollten diese vorher umbenennen.

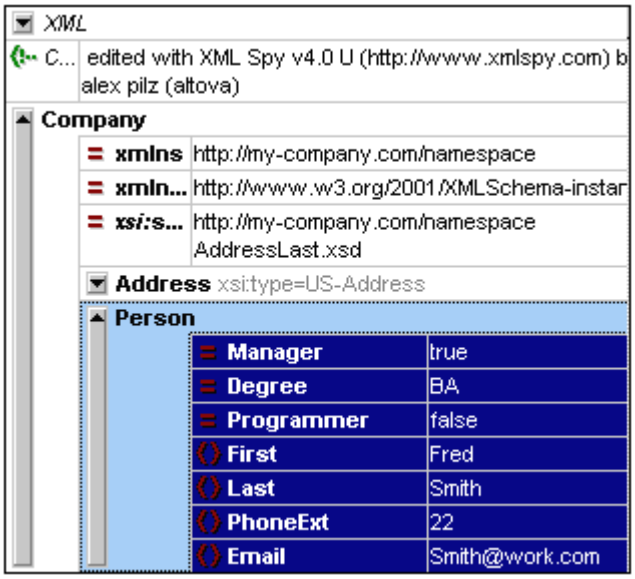
Bitte beachten Sie: Ein XML-Dokument kann auch abgespeichert werden wenn es nicht gültig ist. Beim Speichern werden Sie gefragt, ob Sie das nicht korrekte Dokument speichern wollen. Mit **Save anyway** wird das Dokument in seinem aktuellen, ungültigen Status gespeichert.

Anhängen von Elementen und Attributen in der Grid-Ansicht

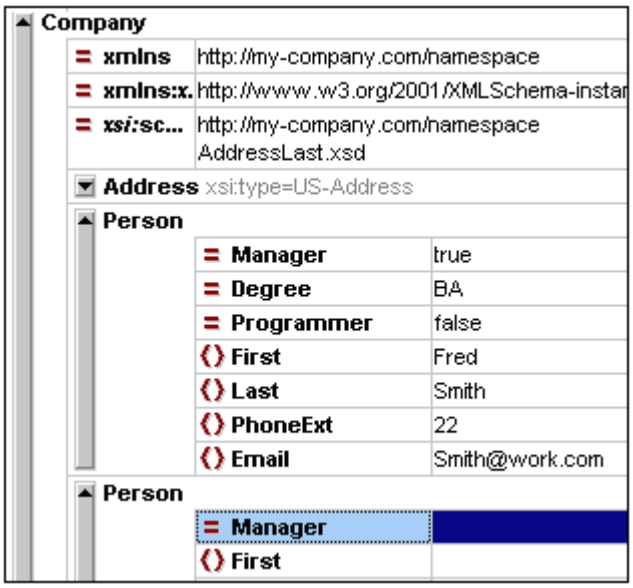
Momentan gibt es im Dokument nur ein `Person`-Element.

So fügen Sie ein neues Element `Person` hinzu:

1. Klicken Sie auf die graue Bildlaufleiste links neben dem Element `Address`, um die `Address`-Elemente auszublenden und mehr Platz in der Ansicht zu schaffen.
2. Wählen Sie das gesamte Element Person, indem Sie in der Grid-Ansicht auf oder unter den Text des Elements `Person` klicken. Beachten Sie, dass das Element `Person` nun in der Element-Eingabehilfe auf dem Register "Anhängen" verfügbar ist.

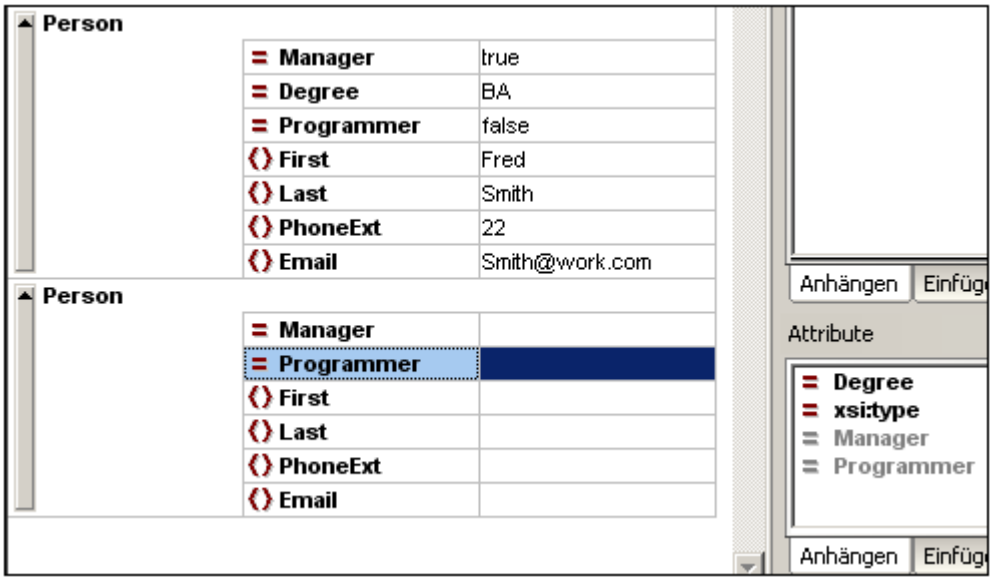


3. Doppelklicken Sie in der Element-Eingabehilfe auf das Element `Person`. Ein neues `Person`-Element sowie alle obligatorischen Child-Elemente werden angehängt (*siehe Abbildung unten*). Beachten Sie, dass das optionale Child-Element `Title` nicht eingefügt wird.



4. Drücken Sie **F12** um in der Grid-Ansicht zum neuen Element `Person` zu wechseln.
5. Klicken Sie auf das Attribut `Manager` im neuen Element `Person`. Beachten Sie die vorhandenen Attribute in der Attribut-Eingabehilfe. Der Eintrag Manager ist ausgegraut, da er bereits verwendet wurde. Werfen Sie auch einen Blick auf das Info-Fenster, in dem nun Informationen über das Attribut `Manager` angezeigt werden.
6. Doppelklicken Sie in der Attribut-Eingabehilfe auf dem Register "Anhängen" auf den Eintrag `Programmer`. Hierdurch wird ein leeres `Programmer`-Attribut hinter dem

`Manager`-Attribut eingefügt.



Das Attribut `Programmer` ist nun in der Attribut-Eingabehilfe ausgegraut.

Sie könnten in dieser Ansicht einen Inhalt für das Element `Person` eingeben. Wechseln wir jedoch nun in die Datenbank/Tabellen-Ansicht der Grid-Ansicht, da sich diese besser zum Editieren einer Struktur mit mehreren Instanzen wie z.B. bei `Person` eignet.

Editieren in der Datenbank/Tabellen-Ansicht

Die Grid-Ansicht enthält eine spezielle Ansicht namens Datenbank/Tabellen-Ansicht (in weiterer Folge Tabellenansicht genannt), die sich zum Bearbeiten von Elementen mit mehreren Instanzen eignet. Einzelne Elementtypen können als Tabelle angezeigt werden. Wenn ein Elementtyp als Tabelle angezeigt wird, werden dessen Children (Attribute und Elemente) als Spalten dargestellt, wobei die einzelnen Instanzen die Zeilen der Tabelle bilden.

Um einen Elementtyp als Tabelle anzuzeigen, wählen Sie eine beliebige Instanz dieses Elementtyps aus und klicken Sie in der Symbolleiste auf das Symbol "Als Tabelle anzeigen" **(XML | Tabelle | Als Tabelle anzeigen**). Elementtypen einer untergeordneten Ebene, die mehrere Instanzen haben, werden ebenfalls in Tabellenform angezeigt. Die Tabellenansicht steht in der Enhanced Grid-Ansicht zur Verfügung und dient zum Bearbeiten jeder Art von XML-Datei (XML, XSD, XSL, usw.).

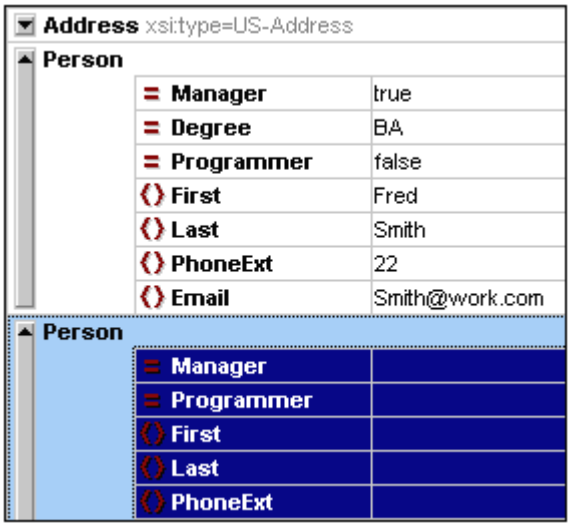
Vorteile der Tabellenansicht
Die Tabellenansicht bietet die folgenden Vorteile:

- Sie können Spalten mit Drag & Drop untereinander austauschen. Im XML-Dokument selbst wird dadurch die relative Position von Child-Elementen oder Attributen für alle Element-Instanzen, die den Zeilen der Tabelle entsprechen, geändert.
- Sortieren von Daten nach Spalten mittels der Funktion **XML | Tabelle | Aufsteigend Sortieren** oder **Absteigend sortieren.**
- Einfügen oder Anhängen von Zeilen mittels der Funktion **XML | Tabelle | Zeile Einfügen**
- Kopieren und Einfügen von **strukturierten Daten** von und in Produkte von Drittanbietern
- Auch die intelligenten Eingabehilfen stehen in der Tabellenansicht zur Verfügung.

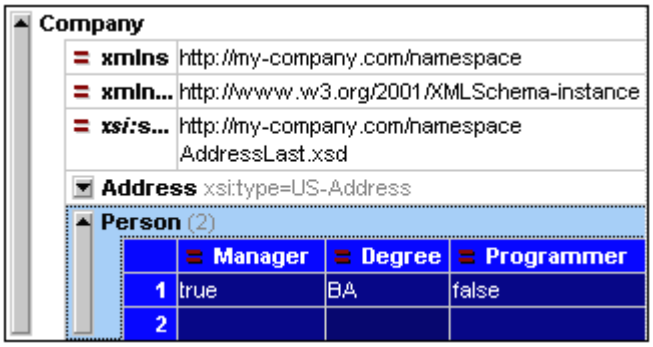
© 2007 Altova GmbH

Anzeigen eines Elementtyps als Tabelle
So zeigen Sie das Element `Person` in Tabellenform an:

1. Wählen Sie in der Grid-Ansicht eines der `Person`-Elemente aus, indem Sie auf oder in die Nähe des `Person`-Texts klicken.



2. Wählen Sie die Menüoption **XML | Tabelle | Als Tabelle anzeigen** oder klicken Sie auf das Symbol . Beide `Person`-Elemente werden in Form einer einzigen Tabellen angezeigt. Die Element- und Attributnamen werden in der Kopfzeile der Tabelle, die einzelnen Elementinstanzen in den Tabellenzeilen angezeigt.

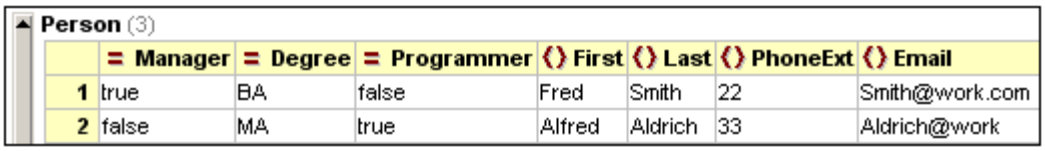


3. Wählen Sie die Menüoption **Ansicht | Optimale Breite** oder klicken Sie auf das Symbol , um die Spaltenbreite zu optimieren.

Bitte beachten Sie: Sie können für einzelne Elementtypen die Tabellenansicht ein- oder ausblenden. Wählen Sie dazu die Tabelle aus (klicken Sie auf den Elementnamen in der Tabelle) und klicken Sie anschließend auf das Symbol "Als Tabelle anzeigen" . Beachten Sie jedoch, dass Child-Elemente, die als Tabellen angezeigt wurden, weiterhin als Tabellen angezeigt werden.

Eingabe von Inhalt in der Tabellenansicht
Um Inhalt für das zweite Person-Element einzugeben, doppelklicken Sie in jede der Tabellenzellen in der zweiten Zeile und geben Sie Daten ein. Beachten Sie jedoch, dass `PhoneExt` eine Ganzzahl bis 99 sein muss, damit die Datei gültig ist. Die intelligenten

Editierfunktionen sind auch innerhalb der Tabellenzellen aktiv, sodass Sie Optionen aus Dropdown-Listen auswählen können, falls Optionen verfügbar sind (Boolscher Inhalt und die Enumerationen für das Attribut `Degree`).



Person (3)

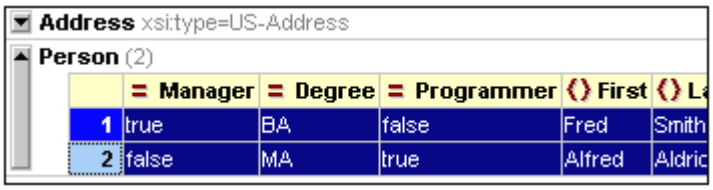
	Manager	Degree	Programmer	First	Last	PhoneExt	Email
1	true	BA	false	Fred	Smith	22	Smith@work.com
2	false	MA	true	Alfred	Aldrich	33	Aldrich@work

Bitte beachten Sie: Die Eingabehilfen sind auch für die als Tabelle dargestellten Elemente und Attribute aktiv. Wenn Sie z.B. in der Element-Eingabehilfe auf den Eintrag `Person` doppelklicken, wird eine neue Tabellenzeile hinzugefügt (D.h. es wird eine neue Instanz zum Element `Person` hinzugefügt).

Kopieren von XML Daten aus und in Drittprodukte

In XMLSpy können Kalkulationstabellendaten aus und in Produkte anderer Anbieter schnell und einfach kopiert werden. Die kopierten Daten können in XMLSpy sowie in Produkten anderer Anbieter in nativem Format der Anwendung verwendet werden. In diesem Abschnitt erfahren Sie, wie man Daten in und aus einem Excel-Datenblatt kopiert. Gehen Sie dazu folgendermaßen vor:

1. Klicken Sie auf die Zeilennummerierung `1`, halten Sie die **Strg**-Taste gedrückt und klicken Sie auf die Zeilennummerierung `2`. Hierdurch werden beide Zeilen markiert.

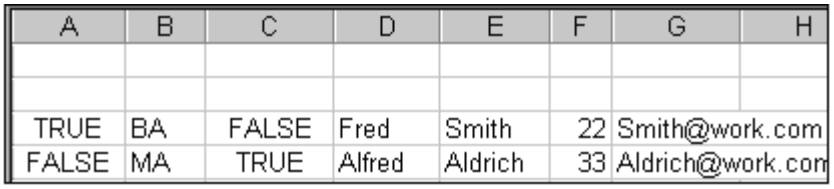


Address xsi:type=US-Address

Person (2)

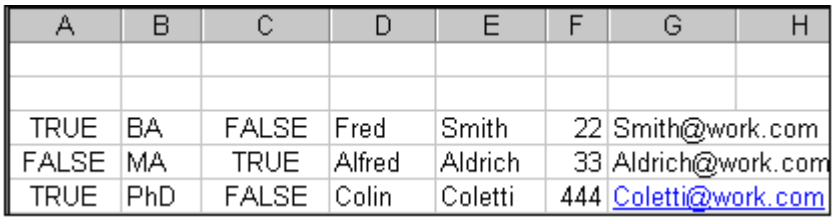
	Manager	Degree	Programmer	First	La
1	true	BA	false	Fred	Smith
2	false	MA	true	Alfred	Aldric

2. Wählen Sie die Menüoption **Bearbeiten | Als strukturierten Text kopieren**. Der Befehl kopiert die Elemente in die Zwischenablage, und zwar genauso, wie sie am Bildschirm erscheinen.
3. Öffnen Sie Excel und fügen Sie die XML Daten (mit **Strg+V**) in ein Excel-Datenblatt ein.



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

4. Fügen Sie eine neue Zeile im Excel-Arbeitsblatt ein und geben Sie eine **dreistellige Zahl** im Element `PhoneExt` ein (z.B. `444`).

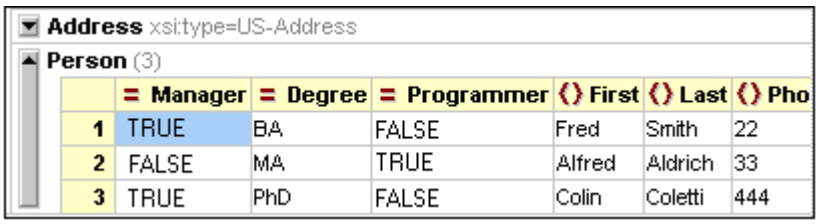


A	B	C	D	E	F	G	H
TRUE	BA	FALSE	Fred	Smith	22	Smith@work.com	
FALSE	MA	TRUE	Alfred	Aldrich	33	Aldrich@work.com	
TRUE	PhD	FALSE	Colin	Coletti	444	Coletti@work.com	

5. Markieren Sie die Daten in Excel, wählen Sie **Bearbeiten | Kopieren**, um die Daten in die Zwischenablage zu kopieren
6. Wechseln Sie wieder zu XMLSpy.

© 2007 Altova GmbH

7. Klicken Sie in XMLSpy in die oberste linke Zelle der Tabelle und wählen Sie **Bearbeiten | Einfügen**.



Address xsi:type=US-Address

Person (3)

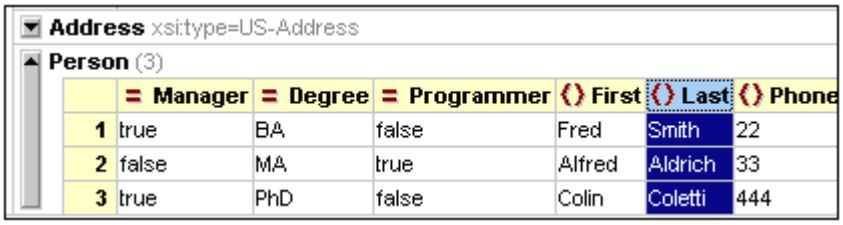
	= Manager	= Degree	= Programmer	{} First	{} Last	{} Pho
1	TRUE	BA	FALSE	Fred	Smith	22
2	FALSE	MA	TRUE	Alfred	Aldrich	33
3	TRUE	PhD	FALSE	Colin	Coletti	444

8. Die aktualisierten Daten wurden in die Tabelle übernommen.
7. Ändern Sie mittels der Menüoption **Suchen | Ersetzen** (Funktionstaste **Strg+H**) die Boolschen Werte `TRUE/FALSE` in `true/false` (in Kleinbuchstaben).

Sortieren der Tabellen nach Spalteninhalt
In der Tabellenansicht können Sie Ihre Daten je nach Spalte in aufsteigender oder absteigender Reihenfolge sortieren. Wir möchten nun die Tabelle `Person` alphabetisch nach Nachnamen sortieren.

So sortieren Sie eine Tabelle nach dem Inhalt einer Spalte:

1. Klicken Sie auf die Kopfzeile in der Spalte **Last**.

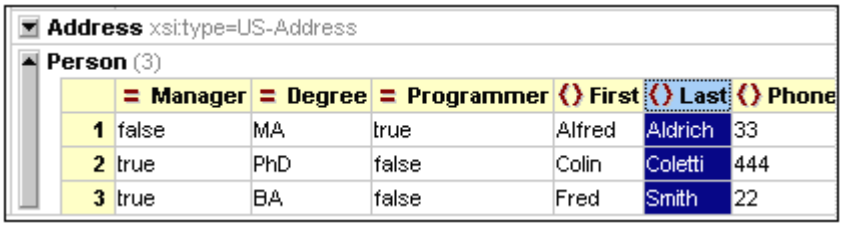


Address xsi:type=US-Address

Person (3)

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

2. Wählen Sie die Menüoption **XML | Tabelle | Aufsteigend sortieren** oder klicken Sie auf das Symbol . Die Spalte wird unter Einbeziehung der gesamten Tabelle alphabetisch sortiert. Die Spalte bleibt hierbei markiert.

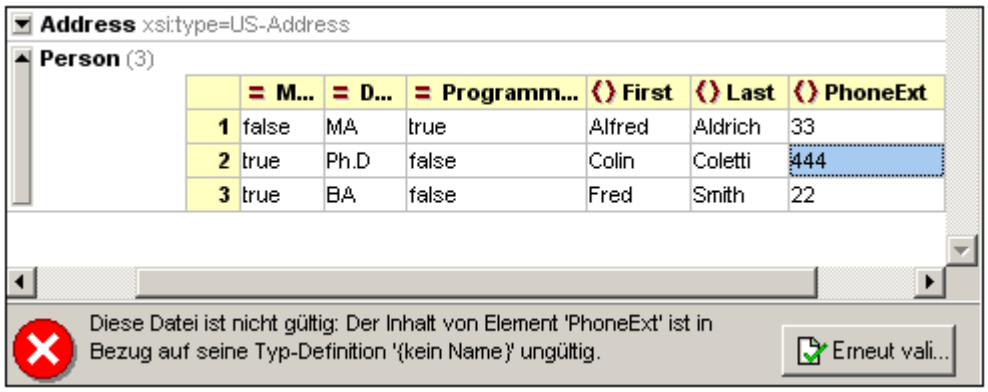


Address xsi:type=US-Address

Person (3)

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

Die Tabelle wird nicht nur in der Ansicht sortiert, sondern die Daten werden auch im zugrundeliegenden XML-Dokument sortiert, d.h. die `Person`-Elemente werden nun alphabetisch nach dem Inhalt der Spalte `Last` sortiert. (Klicken Sie auf das Register "Text", um die Änderungen in der Textansicht zu sehen).
3. Wählen Sie die Menüoption **XML | Validieren** oder drücken Sie die Funktionstaste **F8**. Es wird eine Fehlermeldung angezeigt, dass der Inhalt eines `PhoneExt`-Elements nicht der `PhoneExt`-Facet `in Bezug auf seine Typ-Definition ungültig ist`" entspricht. Das fehlerverursachende PhoneExt-Element erscheint markiert.



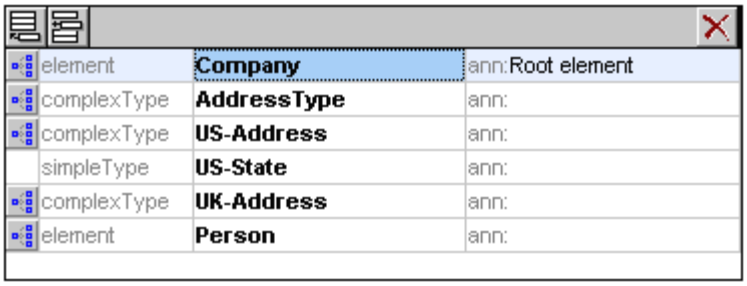
Da diese Durchwahlnummer sich außerhalb des von uns festgelegten Wertebereichs für `PhoneExt` befindet, müssen wir das XML-Schema ändern, damit die Nummer gültig ist. Dies wird im nächsten Abschnitt erläutert.

Ändern des Schemas

Wir haben nun gesehen, dass zweistellige Telefondurchwahlnummern nicht ausreichend sind. Wir werden daher den Wertebereich gültiger Nummern auf drei Stellen erweitern. Sie können das zugrunde liegende XML-Schema öffnen und ändern, ohne Ihr XML-Dokument schließen zu müssen.

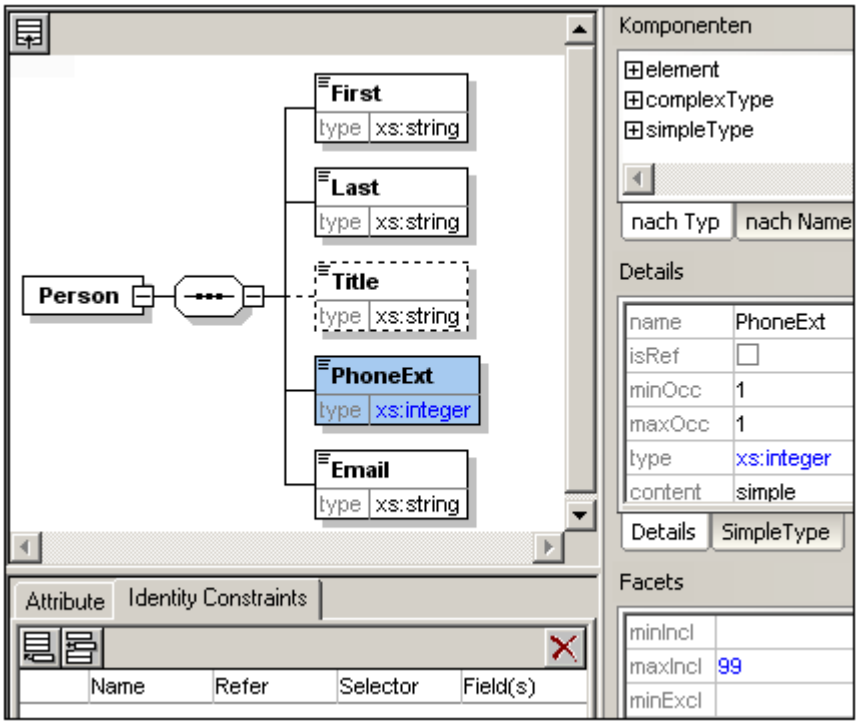
Gehen Sie folgendermaßen vor:

1. Wählen Sie die Menüoption **DTD/Schema | Gehe zu Definition** oder klicken Sie auf das Symbol . Das assoziierte Schemadokument (in diesem Fall `AddressLast.xsd`) wird geöffnet. Wechseln Sie in die Schema/WSDL-Ansicht (siehe Abbildung unten).

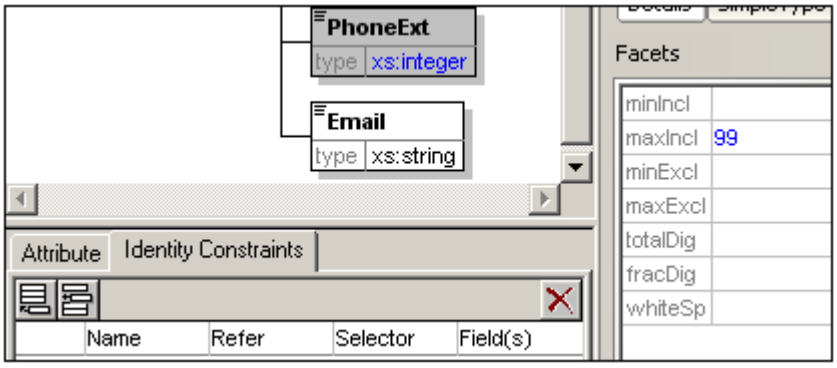


2. Klicken Sie auf das Symbol "Diagramm anzeigen" des globalen Elementes `Person` und klicken anschließend auf das Element `PhoneExt`. Die Facet-Daten werden auf dem Register "Facets" angezeigt.

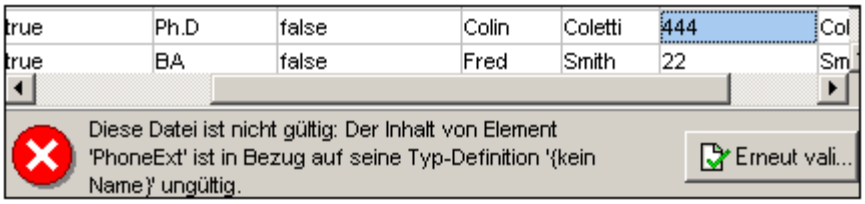
© 2007 Altova GmbH



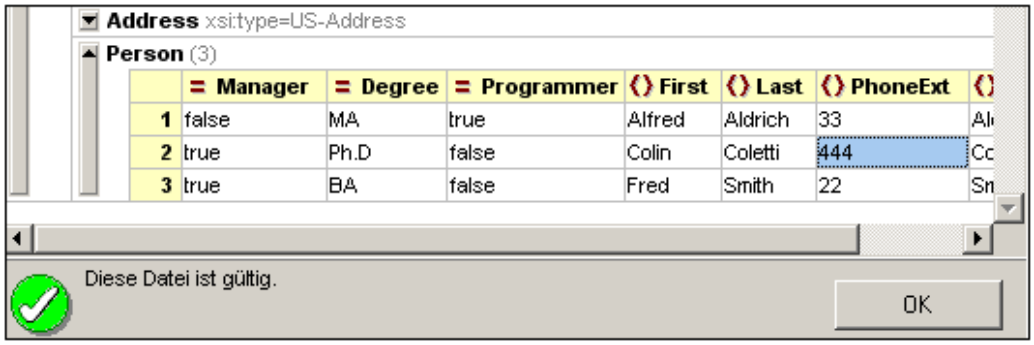
3. Doppelklicken Sie auf dem Register "Facets" auf das Wertefeld `maxIncl`, geben Sie `999` als Maximalwert ein und drücken Sie die Eingabetaste.



4. Speichern Sie das Schema-Dokument.
5. Mit **Strg+TAB** kommen Sie zurück zum XML-Dokument.



6. Klicken Sie auf die Schaltfläche **Revalidate,** um das XML-Dokument erneut zu validieren.



Die Meldung "`Diese Datei ist gültig`" wird angezeigt. Das XML-Dokument ist nun mit der geänderten Schemadefinition konform.

7. Wählen Sie die Menüoption **Datei | Speichern unter...** und benennen Sie die Datei `CompanyLast.xml`. (Denken Sie daran, das Originaldokument `CompanyLast.xml` umzubenennen z.B. in `CompanyLast_orig.xml`).

Bitte beachten Sie: Das Originaldokument `CompanyLast.xml` ist im Verzeichnis `Tutorial` abgelegt.

2.1.6 Transformieren von XML mit XSLT

Ihr Ziel in diesem Abschnitt
Erstellen einer HTML-Datei von einer XML-Datei mittels eines XSLT Stylesheet zum Transformieren der XML-Datei. Beachten Sie, dass die XML-Datei bei einer "Transformation" nicht geändert wird. Es wird nur eine neue Ausgabedatei erzeugt. Das Wort "Transformation" ist eine Konvention.

Methode:
Gehen Sie beim Transformieren folgendermaßen vor:

- Weisen Sie dem XML-Dokument eine vordefinierte XSL-Datei, nämlich `company.xsl` zu.
- Führen Sie die Transformation mit Hilfe eines der beiden in XMLSpy integrierten Altova XSLT-Prozessoren durch (*siehe Hinweis unten*).

In diesem Abschnitt verwendete Befehle
Die folgenden XMLSpy-Befehle werden in diesem Abschnitt verwendet:

XSL/XQuery | XSL zuweisen weist dem aktiven XML-Dokument eine XSL-Datei zu.

XSL/XQuery | Gehe zu XSL öffnet die vom aktiven XML-Dokument referenzierte XSL-Datei.

XSL/XQuery | XSL-Transformation (F10) oder die Symbolleistenschaltfläche transformiert das aktive XML-Dokument mit Hilfe des diesem zugewiesenen XSL Stylesheet. Wenn keine XSL-Datei zugewiesen wurde, werden Sie beim Aufrufen dieses Befehls aufgefordert, eine zuzuweisen.

Bitte beachten Sie: XMLSpy verfügt über zwei integrierte XSLT-Prozessoren, den Altova XSLT 1.0-Prozessor und den Altova XSLT 2.0-Prozessor. Der Altova XSLT 1.0-Prozessor dient zum Transformieren von XSLT 1.0 Stylesheets, während der Altova XSLT 2.0-Prozessor für die Verarbeitung von XSLT 2.0 Stylesheets verwendet wird. Der richtige Prozessor wird von

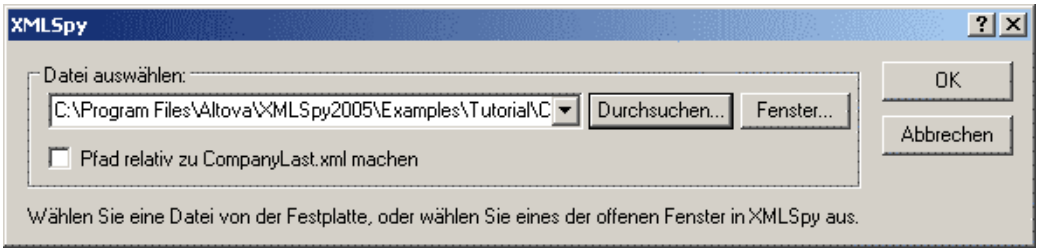
 © 2007 Altova GmbH

XMLSpy automatisch auf Basis des Versionsattributs im Element `xsl:stylesheet` oder `xsl:transform` ausgewählt. In diesem Tutorial verwenden wir für die Transformation XSLT 1.0 Stylesheets. Bei Aufruf des Befehls **XSL-Transformation** wird bei diesen Stylesheets automatisch der Altova XSLT 1.0-Prozessor verwendet.

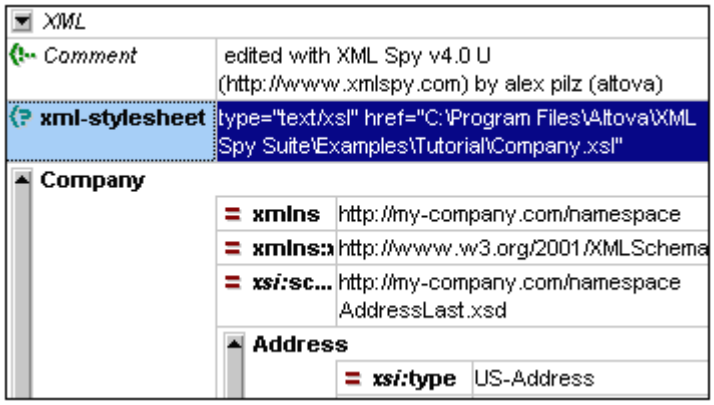
Zuweisen einer XSL-Datei

So weisen Sie der Datei `CompanyLast.xml` eine XSL-Datei zu:

1. Klicken Sie im Hauptfenster auf das Register `CompanyLast.xml`, um es aktiv zu setzen.
2. Wählen Sie die Menüoption **XSL/XQuery | XSL zuweisen**.
3. Klicken Sie auf die Durchsuchen-Schaltfläche und wählen Sie im Ordner `Tutorial` die Datei `Company.xsl` aus. Sie können im Dialogfeld die Option "Pfad relativ zu CompanyLast.xml machen", aktivieren, um den Pfad zur XSL-Datei (im XML-Dokument) relativ zu machen.



4. Klicken Sie auf OK um dem XML-Dokument die XSL-Datei zuzuweisen.



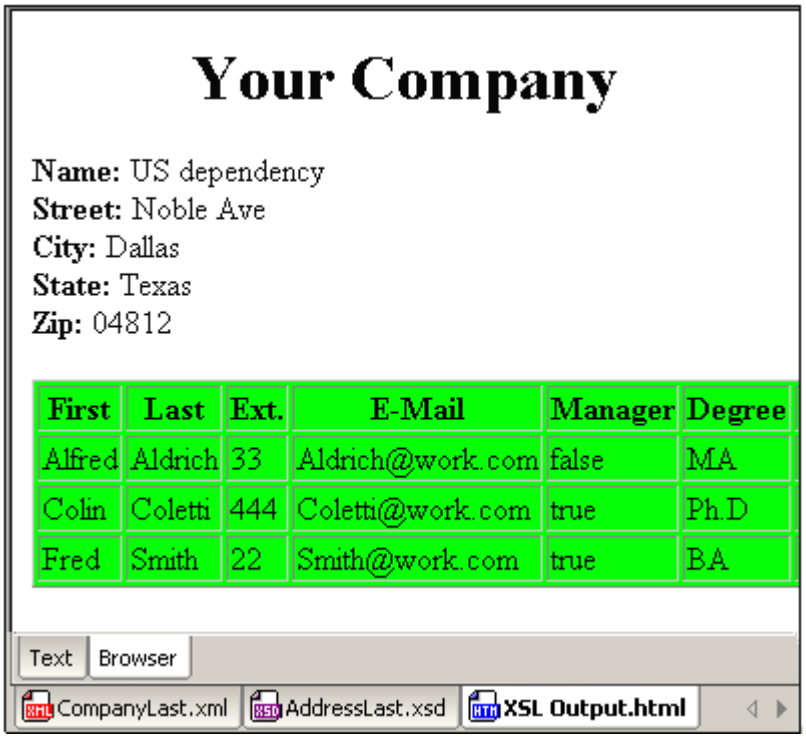
Im XML-Dokument wird eine `XML-stylesheet` Verarbeitungsanweisung eingefügt, die die XSL-Datei referenziert. Wenn Sie das Kontrollkästchen "Pfad relativ zu CompanyLast.mxl machen" aktiviert haben, ist der Pfad relativ; andernfalls ist er absolut (wie in der obigen Abbildung).

Transformieren der XML-Datei

Um das XML-Dokument mit Hilfe der zugewiesenen XSL-Datei zu transformieren, gehen Sie folgendermaßen vor:

1. Stellen Sie sicher, dass die XML-Datei das aktive Dokument ist.
2. Wählen Sie die Menüoption **XSL/XQuery | XSL-Transformation (F10)** oder klicken Sie

auf das Symbol , um die Transformation mittels des in dem XML-Dokument referenzierten XSL Stylesheet zu starten. (Da die Datei `Company.xsl` ein XSLT 1.0-Dokument ist, wird für die Transformation automatisch der integrierte Altova XSLT 1.0-Prozessor ausgewählt). Das Ausgabedokument wird in der Browser-Ansicht angezeigt. Es hat den Namen `XSL Output.html`. (Wenn die HTML-Ausgabedatei nicht generiert wird, stellen Sie sicher, dass die Standarddateierweiterung im Dialogfeld "Optionen" (**Extras | Optionen**) auf dem Register "XSL" auf `.html` gesetzt wurde.) Die Firmendaten werden in einem Block links oben angezeigt, während die Personendaten unterhalb davon in Tabellenform dargestellt werden.



Bitte beachten Sie: Falls in der Ausgabedatei nur eine Tabellenkopfzeile aber keine Tabellendaten zu sehen sind, stellen Sie bitte sicher, dass Sie den Target Namespace für Ihr Schema wie am Beginn des Tutorials unter Definieren von Namespaces beschrieben, definiert haben. Der Namespace muss in allen drei Dateien (Schema, XML und XSL) **identisch** sein.

Ändern der XSL-Datei

Sie können die Ausgabe der Datei ändern, indem Sie die zugrunde liegende XSL-Datei bearbeiten. In diesem Fall soll die Hintergrundfarbe der Tabelle von "Lime (Hellgrün)" in "Yellow (Gelb)" geändert werden.

Gehen Sie folgendermaßen vor:

1. Klicken Sie auf das Register `CompanyLast.xml` um es aktiv zu setzen und stellen Sie sicher, dass Sie sich in der Grid-Ansicht befinden.
2. Wählen Sie die Menüoption **XSL/XQuery | Gehe zu XSL**.

© 2007 Altova GmbH

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:xsi ="http://www.w3.org/2000/10/XMLSchema-instance"
xmlns:my="http://my-company.com/namespace">

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

Mit diesem Befehl wird die Datei `Company.XSL` geöffnet, die in der XML-Datei referenziert wird.

3. Gehen Sie zur Zeile `<table border="1" bgcolor="lime">` und ändern Sie den Eintrag von `bgcolor="lime"` in `bgcolor="yellow"`.

```
<h1><center>Your Company</center></h1>
<xsl:apply-templates select="//my:Address"/>
    <table border="1" bgcolor="yellow">
        <thead align="center">
            <td><strong>First</strong></td>
            <td><strong>Last</strong></td>
```

4. Wählen Sie die Menüoption **Datei | Speichern,** um die Änderungen an der XSL-Datei zu speichern.
5. Klicken Sie auf das Register `CompanyLast.xml` um die XML-Datei aktiv zu setzen, und wählen Sie **XSL/XQuery | XSL-Transformation** oder drücken Sie die Funktionstaste **F10**. In der Browser-Ansicht von XMLSpy erscheint eine neue Datei namens `XSL Output.html`. Die Hintergrundfarbe der Tabelle ist gelb.

Your Company

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

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

6. Wählen Sie die Menüoption **Datei | Speichern** und benennen Sie die Datei `Company.html`.

2.1.7 Projektverwaltung

In diesem Abschnitt werden Sie mit den Projektverwaltungsfunktionen von XMLSpy vertraut gemacht. Nachdem Sie die Vorteile der Gliederung von XML-Dateien in Projekten kennengelernt haben, werden wir die soeben erstellten Dateien in einem einfachen Projekt strukturieren.

Vorteile von Projekten

Im Folgenden sind die Vorteile der Strukturierung von XML-Dateien in Form von Projekten aufgelistet.

- Dateien und URLs können durch gemeinsame Erweiterungen oder andere Kriterien in Ordner gruppiert werden.
- Batchprozesse können bestimmten Ordnern oder dem ganzen Projekt zugewiesen werden.
- DTD / XML-Schemas können bestimmten Ordnern zugewiesen werden, um eine sofortige Validierung der darin befindlichen Dateien zu ermöglichen.
- XSL-Dateien können bestimmten Ordnern zugewiesen werden, um eine sofortige Transformation der darin befindlichen Dateien zu erlauben.
- Es können Zielordner für die XSL-Transformationsdateien definiert werden.

Alle Einstellungen können unter **Projekt | Projekteigenschaften...** definiert werden. Im nächsten Abschnitt werden Sie über das Menü "Projekt" ein Projekt erstellen.

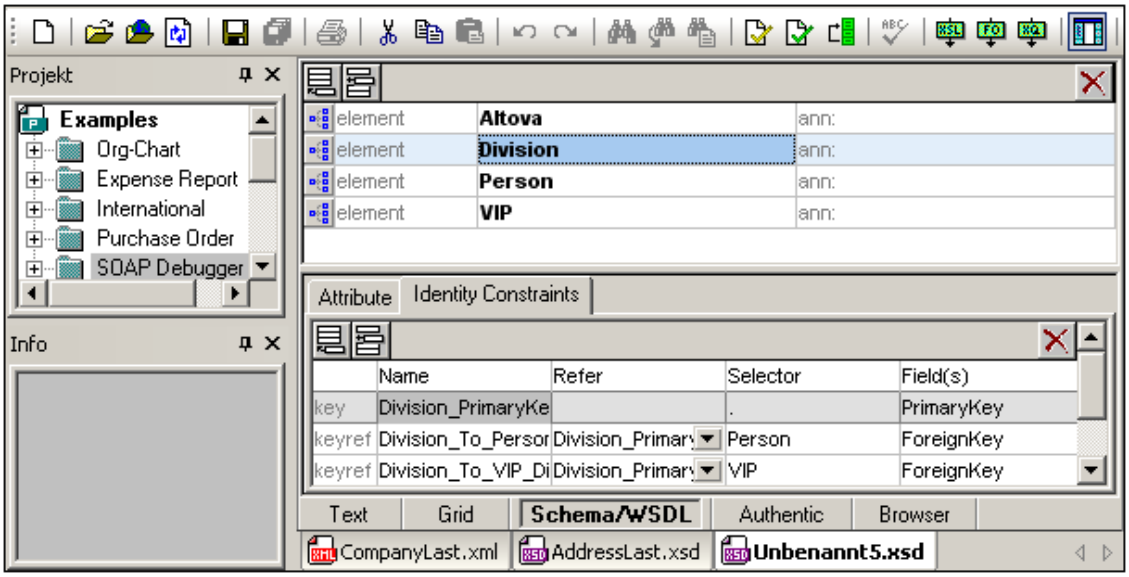
Des Weiteren stehen die folgenden Projektfunktionen zur Verfügung:

- XML-Dateien können unter Versionskontrolle gestellt werden (**Projekt | Versionskontrolle | Zu Versionskontrolle hinzufügen...**). Weitere Informationen erhalten Sie im Abschnitt Versionskontrolle der Online-Hilfe.
- Persönliche, netzinterne und Webordner können in Projekte eingefügt werden, wodurch die Batchvalidierung ermöglicht wird.

© 2007 Altova GmbH

Erstellen eines Projekts

Im Hauptfenster sind nun viele verschiedene Dateien offen.

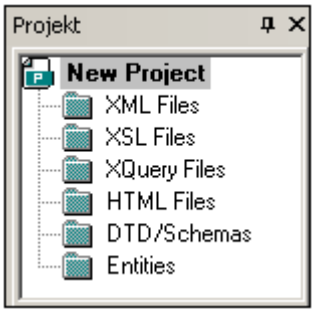


Sie können diese Dateien in einem Tutorial-Projekt gliedern. Zuerst müssen Sie ein neues Projekt erstellen und anschließend die Tutorial-Dateien zu den jeweiligen Unterordnern des Projekts hinzufügen.

Erstellen eines einfachen Projekts

Um ein einfaches Projekt zu erstellen, gehen Sie folgendermaßen vor:

1. Wählen Sie die Menüoption **Projekt | Neu**. Im Projektfenster wird ein Projektordner mit dem Namen `New Project` erstellt. Das neue Projekt enthält leere Ordner für typische XML-Dateikategorien in einem Projekt (*siehe Abbildung unten*).



2. Klicken Sie auf das Register `CompanyLast.xml` um diese Datei im Hauptfenster aktiv zu setzen.
3. Wählen Sie die Menüoption **Projekt | Aktive und verwandte Dateien hinzufügen**. Es werden zwei Dateien zum Projekt hinzugefügt: `CompanyLast.xml` und `AddressLast.xsd`. Beachten Sie, dass durch Verarbeitungsanweisungen referenzierte Dateien (wie z.B. XSLT-Dateien) nicht als verwandte Dateien gelten.
4. Wählen Sie die Menüoption **Projekt | Projekt speichern** und speichern Sie das Projekt unter dem Namen `Tutorial`.

Hinzufügen von Dateien zu einem Projekt

Sie können auch andere Dateien zum Projekt hinzufügen. Gehen Sie dazu folgendermaßen vor:

1. Klicken Sie auf eine beliebige offene XML-Datei (mit der Erweiterung `.xml`) mit Ausnahme von `CompanyLast.xml`, um diese XML-Datei zur aktiven zu machen. (Wenn keine andere XML-Datei geöffnet ist, öffnen Sie eine oder erstellen Sie eine neue XML-Datei).
2. Wählen Sie die Menüoption **Projekt | Aktive Datei hinzufügen.** Die XML-Datei wird anhand ihres Dateityps (.xml) in den Ordner XML Files hinzugefügt.
3. Fügen Sie nun auf dieselbe Art eine HTML-Datei und eine XSD-Datei (z.B. `Company.html` und `AddressFirst.xsd`) zum Projekt hinzu. Diese Dateien werden zum Ordner HTML Files bzw. DTD/Schemas hinzugefügt.
4. Speichern Sie das Projekt, entweder durch Auswahl der Menüoption **Projekt | Speichern** oder durch Auswahl einer beliebigen Datei oder eines Ordners im Projektfenster und Klicken auf das Symbol "Speichern" in der Symbolleiste (oder **Datei | Speichern**).

Bitte beachten Sie: Alternativ dazu können Sie mit der rechten Maustaste auf einen Projektordner klicken und den Befehl "Aktive Datei hinzufügen" auswählen, um die aktive Datei zu diesem bestimmten Ordner hinzuzufügen.

Andere nützliche Befehle
Hier sind einige weitere gebräuchliche Projektbefehle aufgelistet:

- Um einen neuen Ordner zu einem Projekt hinzuzufügen, wählen Sie die Menüoption **Projekt | Projektordner in Projekt einfügen** und fügen Sie den Namen des Projektordners ein.
- Um einen Ordner aus einem Projekt zu löschen, rechtsklicken Sie auf den Ordner und wählen Sie im Kontextmenü den Befehl **Löschen**.
  Um eine Datei aus einem Projekt zu löschen, wählen Sie die Datei aus und drücken Sie die **Entf**-Taste.

2.1.8 Fertig!

Herzlichen Glückwunsch!

Wir hoffen, dass Ihnen diese Anleitung geholfen hat, um die Grundzüge von XMLSpy kennen zu lernen. Weitere Informationen finden Sie in der kontextsensitiven Online-Hilfe. Oder drucken Sie die PDF-Version des Tutorials aus, die sich im Anwendungsordner von XMLSpy befindet (tutorial.pdf).

© 2007 Altova GmbH

2.2 Tutorial Authentic-Ansicht

In der Authentic-Ansicht können Sie XML-Dokumente über eine grafische WYSIWYG-Oberfläche (*Abbildung unten*) wie in einer Textverarbeitungsanwendung, z.B. Microsoft Word, bearbeiten. Sie müssen dabei nur die Daten eingeben und sich nicht um die Formatierung des Dokuments kümmern, da die Formatierung bereits im Stylesheet definiert ist, das festlegt, wie das XML-Dokument in der Authentic-Ansicht dargestellt wird. Das Stylesheet (StyleVision Power Stylesheet, in diesem Tutorial als SPS bezeichnet) wird im Produkt Altova StyleVision erstellt:

Ein XML-Dokument kann in der Authentic in zweierlei Hinsicht bearbeitet werden: (i) Sie können die Struktur des Dokuments bearbeiten (z.B. Dokumentteile wie z.B. Absätze oder Überschriften hinzufügen oder löschen), und (ii) Sie können Daten eingeben (den Inhalt der Dokumentteile).

Dieses Tutorial enthält eine schrittweise Anleitung zu Folgendem:

- Öffnen einer bestehenden XML-Datei in der Authentic-Ansicht. Damit ein Dokument in der Authentic -Ansicht bearbeitet werden kann, mit es mit einer SPS-Datei verknüpft sein.
- Aussehen der Benutzeroberfläche der Authentic-Ansicht und eine allgemeine Beschreibung der wichtigsten Bearbeitungsfunktionen.
- Bearbeiten der Dokumentstruktur durch Einfügen und Löschen von Nodes.
- Eingabe von Daten in das XML-Dokument.
- Eingabe von Attributwerten über die Eingabehilfe "Attribute" und die Entity-Werte
- Drucken des Dokuments

Denken Sie daran, dass dieses Tutorial als Einführung gedacht ist und daher nicht auf komplizierte Aspekte eingeht. Zusätzliches Referenzmaterial und nähere Informationen zu einzelnen Funktionen finden Sie in den Abschnitten zur Oberfläche der Authentic-Ansicht.

Voraussetzungen für das Tutorial
Sie finden alle für das Tutorial benötigten **Dateien** im Ordner `XMLSpy2007\Examples` des Altova-Applikationsordners. Es sind die Dateien:

- `NanonullOrg.xml` (das XML-Dokument, das Sie öffnen)
- `NanonullOrg.sps` (das StyleVision Power Stylesheet, mit dem das XML-Dokument verknüpft ist)
- `NanonullOrg.xsd` (das XML-Schema, auf dem das XML-Dokument und das StyleVision Power Stylesheet basieren und mit dem diese verknüpft sind)
- `nanonull.gif` und `Altova_right_300.gif` (zwei Bilddateien, die in dem Tutorial verwendet werden)

Bitte beachten Sie: An einigen Stellen des Tutorials werden Sie gebeten, sich den XML-Text des XML-Dokuments anzusehen (also nicht die Authentic-Ansicht des Dokuments). Wenn die von Ihnen verwendete Altova Produktversion keine Textansicht enthält (dies ist bei der kostenlosen Authentic Desktop und der Authentic Browser der Fall), verwenden Sie einen einfachen **Texteditor**, wie Wordpad oder Notepad, um den Text des XML-Dokuments anzuzeigen.

Vorsicht: Verwenden Sie für das Tutorial am besten eine Kopie der Datei `NanonullOrg.xml`, damit Sie das Original im gegebenen Fall jederzeit wieder aufrufen können.

2.2.1 Öffnen eines XML-Dokuments in der Authentic-Ansicht

Sie können in der Authentic-Ansicht ein vorhandenes XML-Dokument bearbeiten oder ein neues XML-Dokument erstellen. In diesem Tutorial werden Sie ein vorhandenes XML-Dokument in der Authentic-Ansicht öffnen (wie in diesem Abschnitt beschrieben) und erfahren, wie Sie es bearbeiten können (Beschreibung siehe nächste Abschnitte). Außerdem erfahren Sie in diesem Abschnitt, wie ein neues XML-Dokument erstellt werden kann, um in der Authentic-Ansicht bearbeitet zu werden.

Öffnen eines vorhandenen XML-Dokuments
Die Datei, die Sie öffnen werden, heipt `NanonullOrg.xml`. Sie befindet sich im Ordner `Examples` der Altova-Applikation. Es gibt zwei Möglichkeiten, um `NanonullOrg.xml` zu öffnen
:

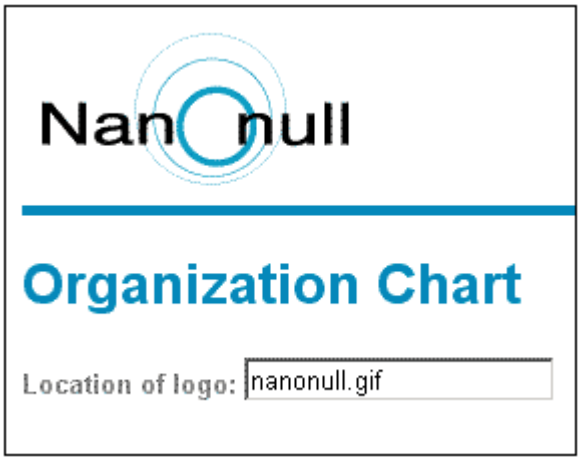
- Klicken Sie in Ihrem Altova-Produkt auf **Datei | Öffnen**, suchen Sie anschließend die Datei `NanonullOrg.xml` im Dialogfeld, das nun angezeigt wird und klicken Sie auf "Öffnen".
- Suchen Sie die Datei im Windows Explorer, rechtsklicken Sie darauf und wählen Sie Ihr Altova-Produkt als die Applikation aus, mit der die Datei geöffnet werden soll.

Die Datei `NanonullOrg.xml` wird direkt in der Authentic-Ansicht (*Abbildung unten*) geöffnet.

- Denn:

1. der Datei wurde bereits ein StyleVision Power Stylesheet (SPS) zugewiesen und.
2. im Dialogfeld "Optionen" (**Extras | Optionen**) auf dem Register "Ansicht" ist die Option, XML-Dateien in der Authentic-Ansicht zu öffnen, wenn eine SPS-Datei zugewiesen wurde, aktiviert. (Andernfalls würde die Datei in der Textansicht geöffnet.)

© 2007 Altova GmbH



Beachten Sie: Wie ein XML-Dokument in der Authentic-Ansicht angezeigt wird, wird im SPS definiert. Ohne ein SPS kann das Dokument nicht in der Authentic-Ansicht dargestellt werden.

Erstellen eines neuen XML-Dokuments auf Basis eines SPS
Sie können auch ein neues XML-Dokument erstellen, das auf einem SPS basiert. Dabei gibt es zwei Möglichkeiten: über den Menübefehl **Datei | Neu** und über den Menübefehl **Authentic | Neues Dokument erstellen**. In beiden Fällen wird ein SPS ausgewählt.

Über Datei | Neu

1. Wählen Sie **Datei | Neu** und wählen Sie im Dialogfeld "Neues Dokument anlegen" XML als den Dateityp der neu zu erstellenden Datei aus.
2. Klicken Sie auf **STYLEVISION Stylesheet auswählen...** und suchen Sie das gewünschte StyleVision Power Stylesheet.

Über Authentic | Neues Dokument erstellen

1. Wählen Sie **Authentic | Neues Dokument.**
2. Wählen Sie im Dialogfeld "Neues Dokument erstellen" das gewünschte SPS aus.

Wenn dem StyleVision Power Stylesheet eine XML-Vorlagendatei zugewiesen wurde, werden die Daten in der XML-Vorlagendatei als Ausgangsdaten der in der Authentic-Ansicht erstellten Vorlage verwendet.

2.2.2 Die Benutzeroberfläche der Authentic-Ansicht

Die Benutzeroberfläche der Authentic-Ansicht besteht aus einem Hauptfenster, in dem Sie die Dokumentdaten eingeben und bearbeiten, und drei Eingabehilfen. Die Bearbeitung eines Dokuments geht ganz einfach. Wenn Sie die Markup-Tags des Dokuments sehen möchten, blenden Sie die Markup-Tags ein. Geben Sie anschließend den Inhalt Ihres Dokuments ein. Um die Dokumentstruktur zu ändern, verwenden Sie entweder das Kontextmenü oder die Eingabehilfe "Elemente".

Anzeige der XML-Node Tags (Dokument-Markup-Tags)
Ein XML-Dokument ist im Grunde eine Hierarchie von Nodes. Z.B.:

```
<DocumentRoot>
   <Person id="ABC001">
      <Name>Alpha Beta</Name>
      <Address>Some Address</Address>
```

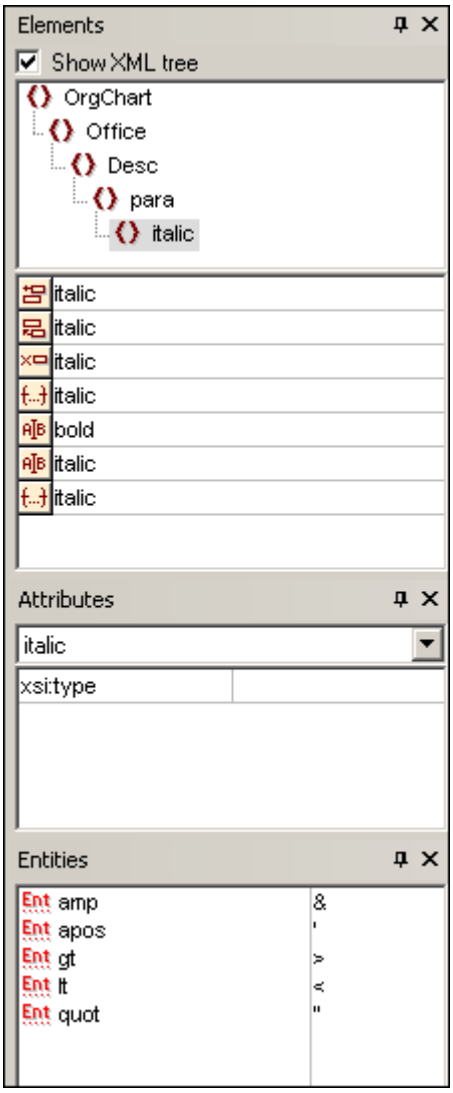
```
        <Tel>1234567</Tel>
    </Person>
</DocumentRoot>
```

Standardmäßig werden die Node-Tags in der Authentic-Ansicht nicht angezeigt. Sie können sie durch Auswahl des Menübefehls **Authentic | Große Markup-Symbole einblenden** (oder der Symbolleisten-Schaltfläche) einblenden. Große Markup-Tags enthalten den Namen des jeweiligen Node. Alternativ dazu können Sie kleine Markup-Symbole (keine Node-Namen in den Tags) und gemischte Markup-Tags (eine Mischung aus großen, kleinen und keinen Markup-Tags, welche durch den Designer des Stylesheets definiert ist; die Standardeinstellung für gemischte Markup-Symbole ist kein Markup) auswählen.

Sie können den Text des XML-Dokuments in der Text-Ansicht Ihres Altova-Produks oder in einem Text Editor ansehen.

Eingabehilfen

Die Benutzeroberfläche bietet drei Eingabehilfen (*Abbildung unten*), die standardmäßig am rechten Rand des Applikationsfensters angeordnet sind. Es sind dies die Eingabehilfen "Element", "Attribute" und "Entities".



 © 2007 Altova GmbH

Element-Eingabehilfe: In der Element-Eingabehilfe werden Elemente angezeigt, die an der aktuellen Cursorposition oder der im Hauptfenster ausgewählten Stelle eingefügt und entfernt werden können. Beachten Sie, dass die Eingabehilfe kontextsensitiv ist; ihr Inhalt ändert sich, je nachdem, wo sich der Cursor gerade befindet, bzw. welcher Text ausgewählt ist. Der Inhalt der Eingabehilfe kann auch auf eine andere Art geändert werden: Wenn in der XML-Struktur ein anderer Node ausgewählt wird, werden in der Eingabehilfe die für diesen Node relevanten Elemente angezeigt. Sie können die Element-Eingabehilfe erweitern, um die XML-Struktur anzuzeigen, indem Sie das Kontrollkästchen "XML-Baumstruktur anzeigen" am oberen Rand der Eingabehilfe aktivieren (*siehe Abbildung oben*). In der XML-Baumstruktur wird die Hierarchie der Nodes vom obersten Element-Node bis hinunter zu dem im Hauptfenster ausgewählten Node angezeigt.

Attribute-Eingabehilfe: In der Attribute-Eingabehilfe werden die Attribute des im Hauptfenster ausgewählten Elements sowie deren Werte angezeigt. Attributwerte können in der Attribute-Eingabehilfe eingegeben oder bearbeitet werden. In der Auswahlliste der Attribute-Eingabehilfe können Element-Nodes von der obersten Elementebene bis hinunter zum ausgewählten Element ausgewählt werden. Bei Auswahl eines Elements aus dieser Dropdown-Liste werden die Attribute dieses Elements in der Eingabehilfe angezeigt, wo sie anschließend bearbeitet werden können.

Entities-Eingabehilfe: Die Entities-Eingabehilfe ist nicht kontextsensitiv. Hier werden alle im Dokument deklarierten Entities angezeigt. Wenn Sie auf eine Entity doppelklicken, wird sie an der Cursorposition eingefügt. Eine Beschreibung, wie man Entites für ein Dokument hinzufügt, finden Sie im Abschnitt Benutzeroberfläche der Authentic-Ansicht.

Kontextmenü
Wenn Sie mit der rechten Maustaste auf eine Stelle in der Authentic-Ansicht doppelklicken, erscheint ein Kontextmenü zum entsprechenden Node. Im Kontextmenü finden Sie Befehle, die Ihnen Folgendes ermöglichen:

- Einfügen von Nodes an dieser Stelle oder vor oder nach dem ausgewählten Node. Das Untermenü enthält eine Liste der Nodes, die an der jeweiligen Stelle eingefügt werden können.
- Entfernen des ausgewählten Node (falls dies im Schema zulässig ist) oder aller entfernbarer übergeordneter Elemente. Die Nodes, die (laut Schema) entfernt werden dürfen, werden im Untermenü aufgelistet.
- Einfügen von Entities und CDATA-Abschnitten. Die für das Dokument deklarierten Entities werden in einem Untermenü aufgelistet. CDATA-Abschnitte können nur innerhalb von Text eingefügt werden.
- Ausschneiden, Kopieren, Einfügen und Löschen von Dokumentinhalt.

Anmerkung: Nähere Informationen über die Benutzeroberfläche finden Sie unter Benutzeroberfläche der Authentic-Ansicht.

2.2.3 Node-Operationen

Es gibt in der Authentic-Ansicht von XML-Dokumenten zwei grundlegende Node-Typen: **Element-Nodes** und **Attribut-Nodes**. Diese Nodes sind alle mit Tags versehen, die Sie einblenden können. Es gibt auch andere Nodes im Dokument, wie z.B. Text-Nodes (die mit keinen Markup-Tags versehen sind) und CDATA-Abschnitt-Nodes (welche mit Markup versehen sind, um sie vom umgebenden Text abzugrenzen).

Die in diesem Abschnitt beschriebenen Node-Operationen beziehen sich nur auf Element-Nodes und Attribut-Nodes. Beim Ausprobieren der hier beschriebenen Operationen sollten Sie

am besten die großen Markup-Symbole einblenden

Anmerkung:

Beachten Sie bitte, dass Sie **nur Elemente derselben oder einer übergeordneten Ebene** vor oder nach dem aktuellen Element einfügen oder anhängen können. Elemente derselben Ebene sind **gleichrangige** Elemente. Bei **gleichrangigen** Elementen eines Absatzelements handelt es sich entweder um andere Absatzelemente oder um Listen, eine Tabelle, eine Grafik usw. handeln. Gleichrangige Elemente können vor oder nach dem Element vorkommen. Elemente übergeordneter Ebenen sind **Vorfahren**-Elemente und Geschwisterelemente von Vorfahren. Bei Vorfahren eines Absatzelements könnte es sich um einen Abschnitt, ein Kapitel, einen Artikel usw. handeln. Ein Absatz in einem gültigen XML-Dokument hat bereits Vorfahren. Wenn Sie daher in der Authentic-Ansicht ein Element einer übergeordneten Ebene erstellen, wird dieses als Geschwisterelement des jeweiligen Vorfahrenelements erstellt. Wenn z.B. ein Abschnittselement an einen Absatz angehängt wird, wird es als Geschwisterelement des Abschnitts erstellt, der das aktuelle Absatzelement enthält - und zwar als der letzte Abschnitt auf dieser Ebene.

Durchführen von Node-Operationen

Node-Operationen können entweder durch Auswahl eines Befehls im Kontextmenü oder durch Klicken auf den Eintrag für die Node-Operation in der Element-Eingabehilfe durchgeführt werden. In einige Fällen kann ein Element oder Attribut hinzugefügt werden, indem Sie in der Authentic-Ansicht des Dokuments auf den Link Node hinzufügen klicken. In speziellen Fällen, in denen Elemente als Absätze oder Listeneinträge definiert wurden, können Sie durch Drücken der Eingabetaste ein neues gleichrangiges Element derselben Art erstellen, wenn sich der Cursor in einem derartigen Element befindet. In diesem Abschnitt erfahren Sie außerdem, wie Nodes über die Funktionen Element anwenden, Node entfernen und Element löschen erstellt und gelöscht werden können.

Einfügen von Elementen

Elemente können an den folgenden Stellen eingefügt werden:

- an der Cursorposition innerhalb eines Element-Node. Die an dieser Stelle verfügbaren Elemente werden im Untermenü des Kontextmenüs **Einfügen** aufgelistet. In der Element-Eingabehilfe werden Elemente, die an dieser Stelle eingefügt werden können, durch ein Symbol markiert. Platzieren Sie im Dokument `NanonullOrg.xml` den Cursor in das Element `para` und erstellen Sie über das Kontextmenü und die Element-Eingabehilfe die Elemente `bold` und `italic`.
- vor oder nach dem ausgewählten Element oder einem seiner Vorfahren, wenn dies laut Schema gestattet ist. Wählen Sie im ersten Untermenü, das angezeigt wird, das Element (ausgewähltes Element oder einen Vorfahren dieses Elements) aus, vor/nach dem das Element eingefügt werden soll. Wählen Sie nun im zweiten Untermenü, das nun angezeigt wird, das gewünschte Element aus, das Sie einfügen möchten. In der Element-Eingabehilfe werden Elemente, die vor oder nach dem ausgewählten Element eingefügt werden können, durch die Symbole bzw. gekennzeichnet. Beachten Sie, dass Sie in der Element-Eingabehilfe Elemente nur vor/nach dem ausgewählten Element einfügen können, nicht aber vor/nach dem Vorfahrenelement. Probieren Sie diesen Befehl aus, indem Sie den Cursor zuerst in das Element `para` und anschließend in die Tabelle mit der Liste der Angestellten platzieren.

Link "Node hinzufügen"

Wenn ein Element oder Attribut im Dokument-Design nicht aber im XML-Dokument selbst vorhanden ist, wird an der Stelle im Dokument, an der dieser Node definiert ist, ein `Node hinzufügen` Link angezeigt. Um diesen Link in der Zeile mit dem Text *Location of logo* zu

 © 2007 Altova GmbH

sehen, wählen Sie den Node `@href` im Element `CompanyLogo` aus und löschen Sie ihn (durch Drücken der **Entf**-Taste). Der Link `@href hinzufügen` wird im Element `CompanyLogo`, das Sie bearbeitet haben, angezeigt (*Abbildung unten*). Wenn Sie auf den Link klicken, wird der Node `@href` zum XML-Dokument hinzugefügt. Das Textfeld innerhalb des Tags `@href` wird angezeigt, weil im Design festgelegt ist, dass der Node `@href` auf diese Art hinzuzufügen ist. Sie müssen nun noch den Wert (oder Inhalt) des Node `@href` eingeben. Geben Sie den Text `nanonull.gif` ein.

Wenn das Contet Model eines Elements nicht eindeutig ist, wenn z.B. festgelegt ist, dass eine Sequenz von Child-Elementen in jeder beliebigen Reihenfolge angezeigt werden kann, so wird der Link `...hinzufügen` angezeigt. Beachten Sie, dass kein Node-Name definiert ist. Wenn Sie auf den Link klicken, erscheint eine Liste von Elementen, die an dieser Stelle eingefügt werden dürfen.

Anmerkung: Der Link "Node hinzufügen" erscheint direkt in der Dokumentvorlage; es gibt keinen entsprechenden Eintrag dazu im Kontextmenü oder in der Element-Eingabehilfe.

Erstellen neuer Elemente mit der Eingabetaste
In Fällen, in denen ein Element (vom Stylesheet Designer) als Absatz oder Listeneintrag formatiert werden muss, wird bei Drücken der Eingabetaste ein neuer Node dieser Art nach dem aktuellen Node eingefügt, wenn sich der Cursor innerhalb eines solchen Node befindet. Sie können diese Funktion im Dokument `NanonullOrg.xml` ausprobieren. Gehen Sie zum Ende des Node `para` (genau vor den End-Tag) und drücken Sie die **Eingabetaste.**

Anwenden von Elementen
In Elementen mit gemischtem Inhalt (denjenigen, die sowohl Text als auch Child-Elemente enthalten) können Sie Textinhalt markieren und darauf eines der zulässigen Child-Elemente anwenden. Der ausgewählte Text wird zum Inhalt des angewendeten Elements. Um Elemente anzuwenden, wählen Sie im Kontextmenü den Befehl "Anwenden" und wählen Sie anschließend eines der vorgeschlagenen Elemente aus. (Wenn auf den ausgewählten Text keine Elemente angewendet werden können, wird der Befehl "Anwenden" im Kontextmenü nicht angezeigt.) In der Element-Eingabehilfe werden Elemente, die für eine Auswahl angewendet werden können, durch das Symbol gekennzeichnet. Wählen Sie im Dokument `NanonullOrg.xml` Text im Element gemischten Inhalts `para` aus und probieren Sie aus, wie sich die Elemente `bold` und `italic` bei Anwendung auswirken.

Der Stylesheet Designer hätte auch eine Symbolleisten-Schaltfläche für die Anwendung eines Elements erstellen können. Im Dokument `NanonullOrg.xml` können die Elemente `bold` und `italic` durch Klicken auf die Symbole "bold" bzw. "italic" in der Authentic-Symbolleiste der Applikation angewendet werden.

Entfernen von Nodes
Ein Node kann entfernt, werden, wenn das Dokument dadurch nicht ungültig wird. Bei Entfernung eines Node werden der Node und sein gesamter Inhalt gelöscht. Sie können einen Node mit dem Kontextmenübefehl **entfernen** entfernen. Wenn der Befehl "Entfernen" markiert wird, erscheint ein Untermenü, das alle Nodes enthält, die entfernt werden dürfen, beginnend beim ausgewählten Node bis zum obersten Node des Dokuments. Um einen zu löschenden Node auszuwählen, platzieren Sie den Cursor in den Node oder markieren Sie den Node (oder einen Teil davon). In der Element-Eingabehilfe werden Nodes, die entfernt werden dürfen, durch das Symbol gekennzeichnet. Sie können einen solchen Node auch entfernen, indem Sie ihn auswählen und die **Entf**-Taste drücken. Probieren Sie im Dokument `NanonullOrg.xml` die oben beschriebenen Methoden aus, indem Sie einige Nodes entfernen. Sie können Ihre Änderungen mit **Strg+Z** rückgängig machen.

Löschen von Elementen
Element-Nodes, bei denen es sich um Child-Elemente mit gemischtem Inhalt (sowohl Text- als auch Element-Children) handelt, können gelöscht werden. Sie können das gesamte Element löschen, wenn Sie den Node auswählen oder den Cursor als Einfügepunkt in den Node setzen. Sie können den Element-Markup-Code eines Textfragments innerhalb des Elements löschen, indem Sie das Textfragment markieren. Klicken Sie nach dem Markieren des Fragments im Kontextmenü auf den Befehl **Löschen** und wählen Sie anschließend das zu löschende Element aus. In der Element-Eingabehilfe werden Elemente, die aus einer bestimmten Auswahl gelöscht werden können, durch das Symbol (Einfügepunktauswahl) und (Bereichsauswahl) gekennzeichnet. Probieren Sie das Löschen von Elementen im Dokument `NanonullOrg.xml` an den Child-Elementen `bold` und `italic` des Elements `para` (welches gemischten Inhalt hat) aus.

Tabellen und Tabellenstruktur
Es gibt zwei Arten von Tabellen in der Authentic-Ansicht:

- *SPS-Tabellen (statische und dynamische)*. Die allgemeine Struktur von SPS-Tabellen wird durch den Stylesheet Designer festgelegt. Innerhalb dieser grundlegenden Struktur sind die einzigen zulässigen Änderungen an der Struktur inhaltsbezogene. So können Sie z. B. neue Zeilen zu einer dynamischen SPS-Tabelle hinzufügen.
- *XML-Tabellen*, in welchen Sie entscheiden, dass der Inhalt eines bestimmten Node (z.B. eines Node für personenspezifische Daten) als Tabelle dargestellt werden soll. Wenn der Stylesheet Designer die Erstellung dieses Node als XML-Tabelle ermöglicht hat, können Sie die Struktur der Tabelle festlegen sowie ihren Inhalt bearbeiten. XML-Tabellen werden im Abschnitt Verwendung von Tabellen in der Authentic-Ansicht näher beschrieben.

2.2.4 Eingabe von Daten in der Authentic-Ansicht

Daten werden im XML-Dokument direkt im Hauptfenster der Authentic-Ansicht eingegeben. Zusätzlich dazu können Daten für Attribute (also der Attributwert) über die Attribute-Eingabehilfe eingegeben werden.Daten werden entweder (i) direkt als Text oder (ii) durch Auswahl einer Option in einem Dateneingabeelement, eingegeben, das anschließend auf einen vordefinierten Texteintrag gemappt wird.

Hinzufügen von Textinhalt
Sie können Elementinhalt und Attributwerte im Hauptfenster der Authentic-Ansicht direkt als Text eingeben. Platzieren Sie dazu den Cursor an die gewünschte Stelle und geben Sie den

 © 2007 Altova GmbH

Text ein. Sie können Text auch aus der Zwischenablage in das Dokument kopieren. Inhalt kann auch mit den normalen Inhaltsbearbeitungsmechanismen wie **Entfernen** und **Feststelltaste,** durch Markieren des Texts und Eingabe des Ersetzungstexts oder durch Löschen des markierten Texts bearbeitet werden.

Um z.B. den Namen der Firma im Feld `Name` des Elements `Office` zu ändern, platzieren Sie den Cursor hinter Nanonull und geben Sie "USA" ein, um den Namen von "Nanonull, Inc." in "Nanonull USA, Inc." zu ändern.

Wenn Text editierbar ist, können Sie den Cursor hineinsetzen und den Text markieren, andernfalls ist das nicht der Fall. Versuchen Sie, einen der **Feldnamen** (nicht den Wert im Feld) "Street", "City", "State/Zip" im Adressblock zu ändern. Sie können den Cursor nicht in diesen Text platzieren, da dies kein XML-Inhalt ist, sondern aus dem StyleVision Power Stylesheet stammt.

Einfügen von Sonderzeichen und Entities
Bei der Eingabe von Daten werden folgende Inhaltsarten speziell behandelt:

- *Sonderzeichen, die als XML-Markup-Symbole verwendet werden* (kaufmännisches Und, Apostroph, Größer als, Kleiner als und Anführungszeichen). Diese Zeichen stehen als vordefinierte Entities zur Verfügung und können durch Doppelklick auf die entsprechende Entity in der Entities-Eingabehilfe im Dokument eingefügt werden. Wenn diese Zeichen häufig vorkommen (z.B. in Programmcodelisten), können Sie innerhalb von CDATA-Abschnitten eingegeben werden. Klicken Sie zum Einfügen eines CDATA-Abschnitts mit der rechten Maustaste an die gewünschte Stelle und wählen Sie im Kontextmenü den Befehl **CDATA-Abschnitt einfügen**. Der XML-Prozessor ignoriert alle Markup-Zeichen innerhalb von CDATA-Abschnitten. Dies bedeutet auch, dass Sonderzeichen innerhalb von CDATA-Abschnitten als das Zeichen selbst und nicht die Entity-Referenz eingegeben werden müssen.
- *Sonderzeichen, die nicht über die Tastatur eingegeben werden können,* sollten durch Kopieren des Zeichens aus der Zeichentabelle Ihres Systems an die gewünschte Stelle im Dokument eingegeben werden.
- *Ein häufig verwendeter Textstring* kann als Entity definiert werden, die in der Entities-Eingabehilfe angezeigt wird. Die Entity wird an der gewünschten Stelle eingefügt. Platzieren Sie dazu den Cursor an die gewünschte Stelle und doppelklicken Sie in der Entities-Eingabehilfe auf die Entity. Diese Funktion eignet sich dazu, um Dokumente auf aktuellem Stand zu halten, da der Wert des Textstrings nur einmal gespeichert ist. Wenn der Wert geändert werden muss, müssen Sie nur die Entity-Definition ändern.

Anmerkung: Wenn Markup-Tags in der Authentic -Ansicht ausgeblendet werden, werden leere Elemente leicht übersehen. Um sicherzustellen, dass Sie kein leeres Element übersehen haben aktivieren Sie ein der Optionen große Markup-Symbole (oder kleine Markup-Symbole) anzeigen.

Probieren Sie die Verwendung der einzelnen oben beschriebenen Arten von Text aus.

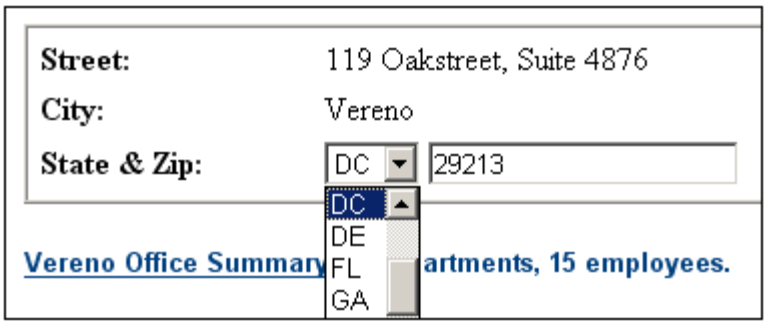
Hinzufügen von Inhalt über ein Dateneingabe-Steuerelement
Beim Editieren von Inhalt wird dieser direkt durch Eingabe des Texts als Elementinhalt

eingegeben. Es gibt aber noch eine andere Methode, um **Elementinhalt** (oder Attributwerte) in der Authentic-Ansicht einzugeben: mit Hilfe von Dateneingabe-Steuerelemente.

Nachfolgend sehen Sie eine Liste von Dateneingabe-Steuerelementen in der Authentic-Ansicht sowie eine Erklärung dazu, wie Daten bei jedem dieser Elemente in die XML-Datei eingegeben werden.

Dateneingabe-Steuerelement	Daten in der XML-Datei
Eingabefeld (Textfeld)	Text wird vom Benutzer eingegeben
Eingabefeld mit mehreren Zeilen	Text wird vom Benutzer eingegeben
Dropdown-Liste	Auswahl des Benutzers wird auf einen Wert gemappt
Kontrollkästchen	Auswahl des Benutzers wird auf einen Wert gemappt
Optionsfeld	Auswahl des Benutzers wird auf einen Wert gemappt
Schaltfläche	Auswahl des Benutzers wird auf einen Wert gemappt

In der statischen Tabelle mit den Adressfeldern (siehe unten) stehen zwei Dateneingabe-Steuerelemente zur Verfügung: Ein Eingabefeld für das Feld "Zip" (PLZ) und eine Dropdown-Liste für das Feld "State" (Bundesstaat). Die Werte, die Sie in die Textfelder eingeben, werden direkt als XML-Inhalt der entsprechenden Elemente eingegeben. Bei anderen Dateneingabe-Steuerelementen wird Ihre Auswahl auf einen Wert gemappt.



Hier sehen Sie den XML-Text für die oben gezeigte Authentic-Ansicht:

```
<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>
```

Beachten Sie, dass die Dropdown-Listen-Auswahl "DC" auf einen Wert "DC" gemappt ist. Der Wert im Feld "zip" wird direkt als Inhalt des Elements `ipo:zip` eingegeben.

© 2007 Altova GmbH

2.2.5 Eingabe von Attributwerten

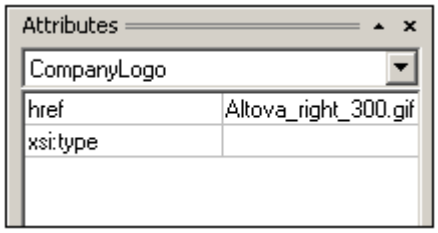
Ein Attribut ist eine Eigenschaft eines Elements. Ein Element kann beliebig viele Attribute haben. Attribute haben Werte. Manchmal müssen XML-Daten als Attributwert eingegeben werden. In der Authentic-Ansicht werden Attributwerte auf zwei Arten eingegeben:

- Als Inhalt im Hauptfenster, wenn das Attribut so definiert wurde, dass es die Eingabe des Werts auf diese Art zulässt.
- In der Attribut-Eingabehilfe

Attributwerte im Hauptfenster
Attributwerte können als Fließtext oder als Text in einem Eingabefeld eingegeben werden oder als vom Benutzer aus einer Auswahl gewählter Eintrag, der auf einen XML-Wert gemappt wird. Attributwerte werden auf dieselbe Art eingegeben, wie Element-Inhalt: siehe Eingabe von Daten in der Authentic-Ansicht. In solchen Fällen erfolgt die Unterscheidung zwischen Element-Inhalt und Attributwerten durch das StyleVision Power Stylesheet und die Daten werden entsprechend behandelt.

Attributwerte in der Attribut-Eingabehilfe
Wenn Sie einen Attributwert eingeben oder ändern wollen, können Sie dies in der Attribut-Eingabehilfe tun. Zuerst wird der Attribut-Node in der Authentic-Ansicht ausgewählt, dann wird der Wert des Attributs in der Attribute-Eingabehilfe eingegeben oder bearbeitet. Der Pfad des in `NanonullOrg.xml` verwendeten Logos wird als der Wert des `href` Attributs des Elements `CompanyLogo` gespeichert. Um ein anderes Logo zu verwenden, gehen Sie vor wie folgt:

1. Wählen Sie das Element `CompanyLogo` aus, indem Sie auf den Tag CompanyLogo klicken. Die Attribute des Elements `CompanyLogo` werden in der Attribut-Eingabehilfe angezeigt.
2. Ändern Sie in der Attribut-Eingabehilfe den Wert des Attributs `href` von `nanonull.gif` in `Altova_right_300.gif` (ein Bild im Ordner `Examples`).



Daraufhin wird das Nanonull-Logo durch das Altova-Logo ersetzt.

Anmerkung: Entities können nicht in der Attribut-Eingabehilfe eingegeben werden.

2.2.6 Hinzufügen von Entities

Bei einer Entity in der Authentic-Ansicht handelt es sich normalerweise (nicht notwendigerweise) um XML-Daten wie z.B. ein einzelnes Zeichen, einen Textstring oder auch ein Fragment eines XML-Dokuments. Auch eine Binärdatei wie z.B. eine Bilddatei kann eine Entity sein. Alle verfügbaren Entities werden in der Entities-Eingabehilfe (*Abbildung unten*) angezeigt. Um eine Entity einzufügen, platzieren Sie den Cursor an die gewünschte Stelle im Dokument und doppelklicken Sie anschließend in der Entities-Eingabehilfe auf die Entity. Beachten Sie, dass Sie Entities nicht in der Attribut-Eingabehilfe eingeben können.

Das kaufmännische Und (&) hat in XML eine spezielle Bedeutung (wie auch das Apostroph, das "Größer als"- und das "Kleiner als"-Zeichen und das doppelte Anführungszeichen). Um diese Zeichen einzufügen, werden Entities verwendet, damit sie nicht mit Zeichen, die eine XML-Bedeutung haben, verwechselt werden. Diese Zeichen sind in der Authentic-Ansicht als Entities verfügbar.

Ändern Sie in `NanonullOrg.xml` den Jobtitel von Joe Martin (in Marketing) in Marketing Manager Europe & Asia. Gehen Sie dazu folgendermaßen vor:

1. Platzieren Sie den Cursor dorthin, wo das kaufmännische Und eingefügt werden soll.
2. Doppelklicken Sie auf die Entity, die als "amp" aufgelistet ist. Daraufhin wird ein kaufmännisches Und eingefügt (*Abbildung unten*)

Anmerkung:
Die Entities-Eingabehilfe ist nicht kontextsensitiv. Es werden immer alle verfügbaren Entities angezeigt, egal wo sich der Cursor gerade befindet. Dies bedeutet nicht, dass eine Entity überall im Dokument eingefügt werden kann. Wenn Sie sich nicht sicher sind, validieren Sie das Dokument nach dem Einfügen der Entity mit: **XML | Validieren** (**F8**).

Definieren Ihrer eigenen Entities
Als Dokument-Bearbeiter können Sie auch Ihre eigenen Entities definieren. Eine Beschreibung dazu finden Sie im Abschnitt Definieren von Entities in der Authentic-Ansicht.

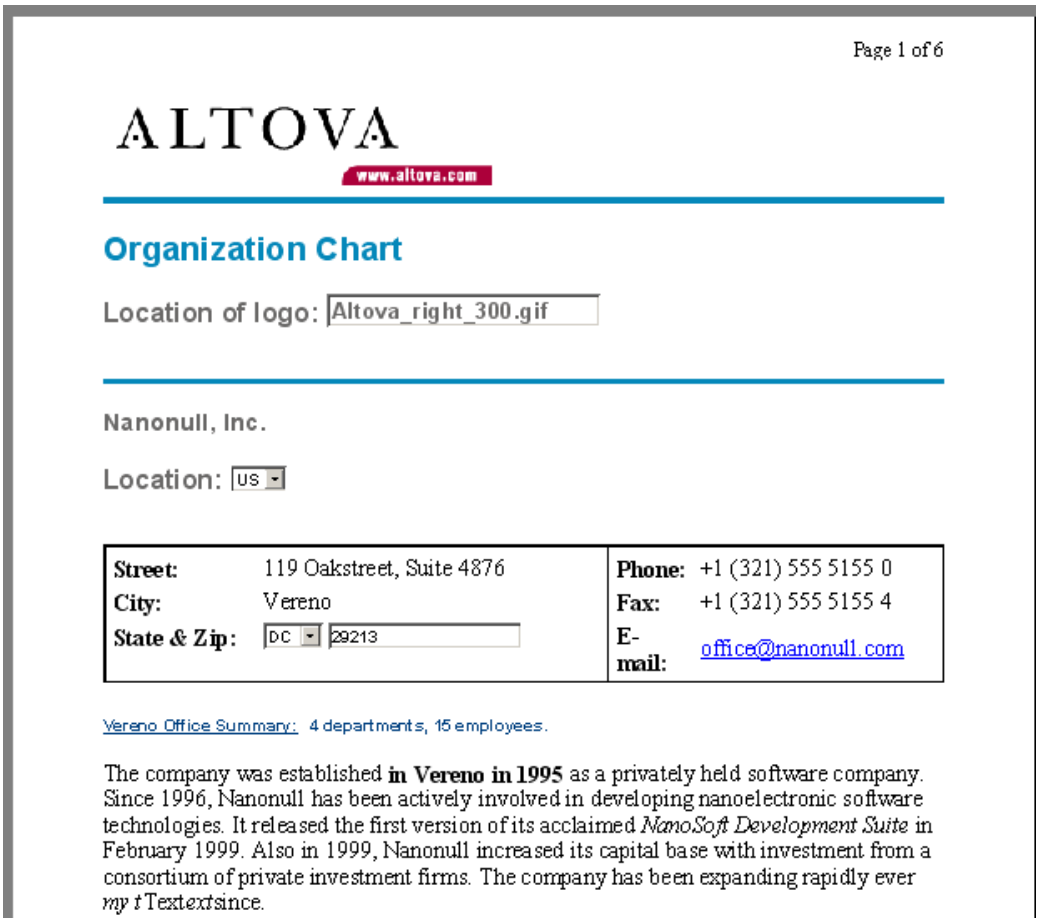
2.2.7 Drucken des Dokuments

Bei Ausdruck eines XML-Dokuments von der Authentic-Ansicht aus bleibt die in der Authentic -Ansicht sichtbare Formatierung erhalten.

So drucken Sie die Datei `NanonullOrg.xml`:

1. Wechseln Sie in den Modus "Markup-Symbole ausblenden", wenn Sie gerade einen anderen Modus verwenden, da sonst auch die Markup-Symbole gedruckt werden.
2. Wählen Sie **Datei | Druckvorschau**, um eine Vorschau aller Seiten anzuzeigen. In der Abbildung unten sehen Sie einen Teil einer um 50 % verkleinerten Seite in der Druckansicht.

 © 2007 Altova GmbH



Beachten Sie, dass die Formatierung auf der Seite dieselbe ist, wie in der Authentic -Ansicht.

3. Um die Datei zu drucken, wählen Sie **Datei | Drucken**.

Beachten Sie, dass Sie auch eine Version des Dokuments drucken können, in der (kleine) Markup-Symbole angezeigt werden. Wechseln Sie dazu in den Modus "Kleine Markup-Symbole einblenden" oder "Große Markup-Symbole einblenden" und wählen Sie den Befehl "Drucken". In beiden Modi werden im Ausdruck kleine Markup-Symbole angezeigt.

2.3 WSDL Tutorial

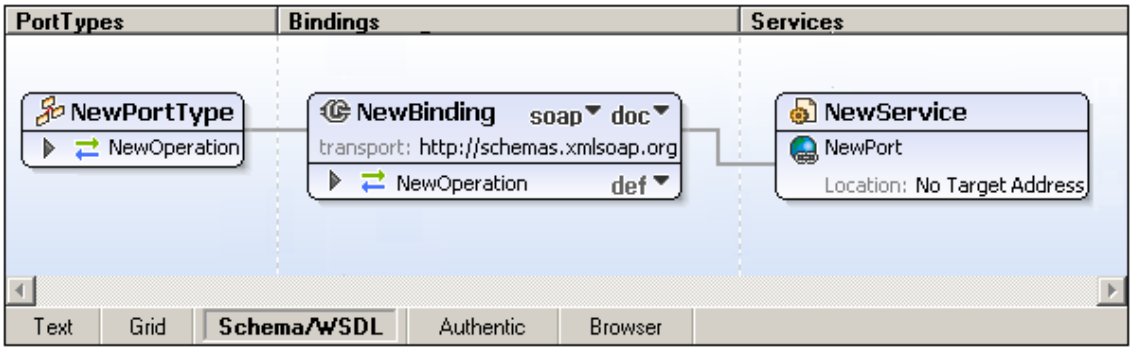
Dieses Tutorial ist in zwei Teile gegliedert:

- eine Beschreibung, wie ein WSDL-Dokument in der grafischen WSDL-Ansicht von XMLSpy erstellt wird. In diesem Teil lernen Sie: (i) wie man mit Hilfe der Menüoption **Datei | Neu** ein rudimentäres WSDL-Dokument erstellt, (ii) wie man einen PortType erstellt (iii) wie man ein Binding erstellt; (iv) wie man einen Service und einen Port erstellt; (v) und die Datei validiert und speichert.
- eine Beschreibung, wie man eine Verbindung zu einem Webservice herstellt, die WSDL-Datei lokal speichert und einen SOAP Request an den Webservice sendet

Sie können das alles in der graphischen WSDL-Ansicht tun und benötigen die Textansicht hierfür nicht. Sie können die WSDL-Komponenten direkt mit Hilfe von Drag & Drop verschieben und Eigenschaftswerte in den Eingabehilfen der WSDL-Ansicht eingeben.

2.3.1 Erstellen eines neuen Dokuments

Um ein neues WSDL-Dokument zu erstellen, wählen Sie den Befehl **Datei | Neu**. Wählen Sie im Dialogfeld "Neues Dokument erstellen", das daraufhin angezeigt wird, `WSDL` als Dokumenttyp aus und klicken Sie auf **OK**. Daraufhin wird das Gerüst für ein neues Dokument erstellt (*Abbildung unten*), welches in der grafischen Schema/WSDL-Ansicht (in diesem Tutorial als WSDL-Ansicht bezeichnet) geöffnet wird.



Zuweisen eines Target Namespace
Wechseln Sie in die Text-Ansicht. Der Start Tag des Elements `wsdl:definitions` sieht ungefähr folgendermaßen aus:

```
1   <?xml version="1.0" encoding="UTF-8"?>
2   <wsdl:definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
3       xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
4       xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
5       xmlns:xs="http://www.w3.org/2001/XMLSchema"
6       xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
7       xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
8       xmlns:tns="http://new.webservice.namespace"
9       targetNamespace="http://new.webservice.namespace">
10
```

Ändern Sie das Attribut des Target Namespace in `http://mywebservice.namespace` oder einen beliebigen anderen Namen (Dieses Tutorial dient schließlich nur dazu, Ihnen zu zeigen, wie ein WSDL-Dokument erstellt wird und stellt keinen tatsächlichen Service bereit). Anschließend sollten Sie auch den Namespace-Wert des `tns` Namespace in `http://mywebservice.namespace` ändern (oder den als Target Namespace gewählten Namespace).

© 2007 Altova GmbH

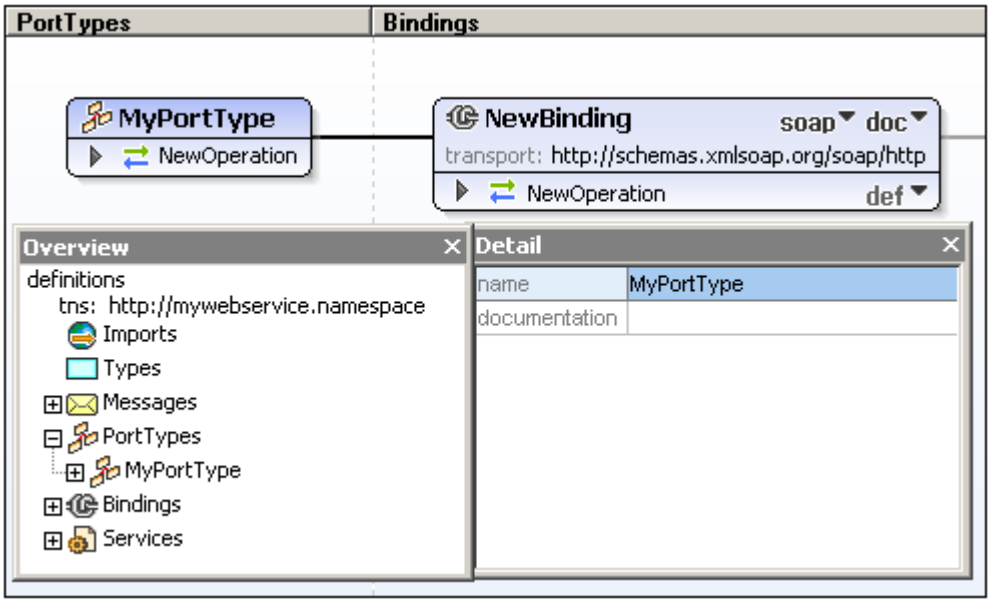
Anmerkung: In den WSDL-Elementen, die im gerade erstellten Dokumentgerüst erstellt wurden, befinden sich WSDL-Elemente im Target Namespace, während Referenzen zu WSDL-Elementen über das Präfix `tns` definiert werden. Beispiel: `<wsdl:binding name="NewBinding" type="tns:NewPortType">`. Damit das `tns` Präfix mit dem Target Namespache übereinstimmt, müssen beide identisch sein.

2.3.2 Erstellen eines PortType

Um einen PortType zu erstellen, sind folgende Schritte erforderlich:

- Benennung des PortType
- Einfügen einer Operation
- Hinzufügen von Input- und Output Messages
- Hinzufügen von Parametern zu Messages

Benennung des PortType
Benennen Sie den PortType `NewPortType` in `MyPortType` um, indem Sie im Design in die Titelleiste des Felds `NewPortType` doppelklicken, den Namen bearbeiten und anschließend die **Eingabetaste** drücken. Beachten Sie, dass sich der Name der PortType auch im Übersichtsfenster und in der Eingabehilfe "Details" ändert (*Abbildung unten*).



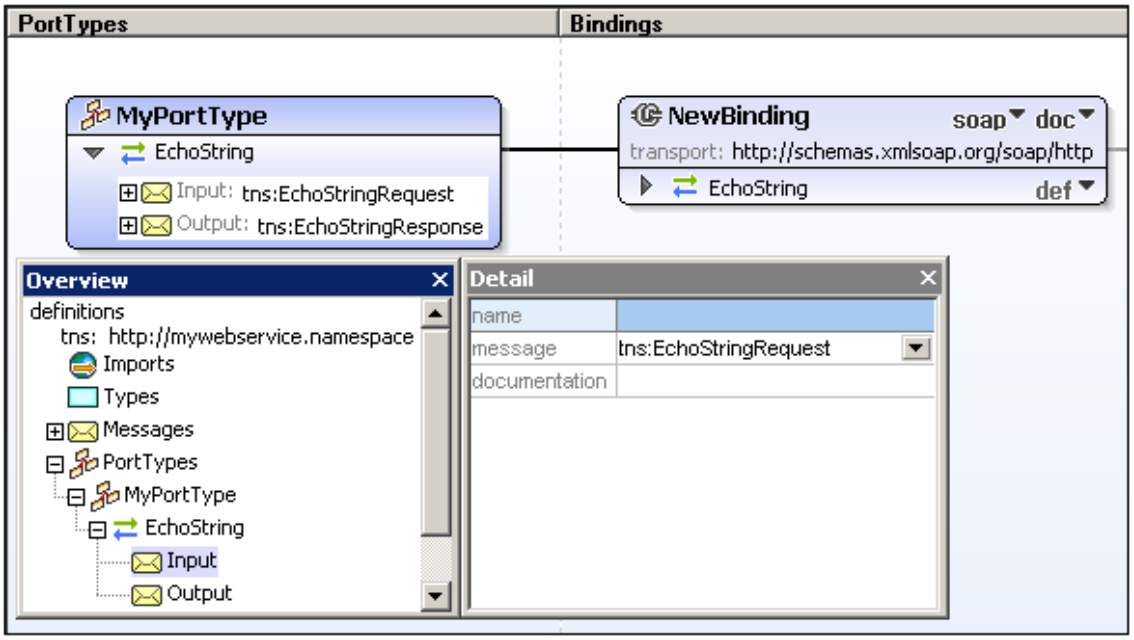
Einfügen einer Operation
Im Fall von `MyPortType` ist eine Operation, `NewOperation`, bereits vorhanden. Daher werden wir mit dieser Operation arbeiten. Zunächst benennen wir `NewOperation` in z.B. `EchoString` um (doppelklicken Sie auf den Namen, bearbeiten Sie ihn und drücken Sie die **Eingabetaste**). (Um zusätzliche Operationen für einen PortType einzufügen, klicken Sie mit der rechten Maustaste auf das PortType-Feld und wählen Sie den Befehl **Operation anhängen**.)

Hinzufügen von Input- und Output Messages
Wenn eine Operation zu einem PortType hinzugefügt wird, können Sie auswählen, welchem von fünf Typen die Operation angehören soll.

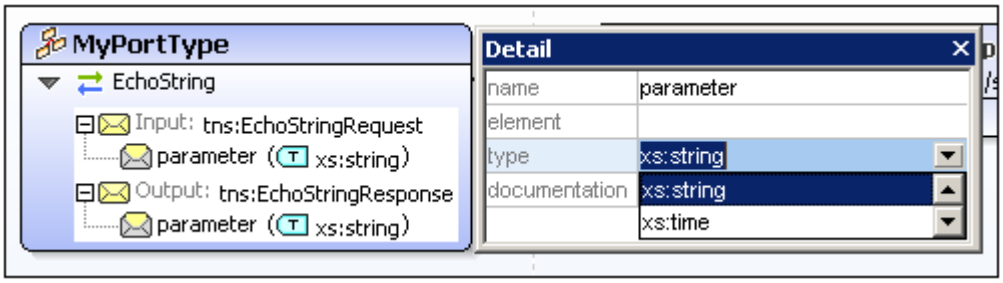
- Request response

- Solicit response
- One-way
- Notification
- Leere Operation

Für jeden Typ werden je nach Operationstyp automatisch Input- und Output Messages hinzugefügt. Bei Auswahl des Befehls **Leere Operation** können Sie mit einem Rechtsklick auf die Operation den einzufügenden Message-Typ auswählen. Eine Message kann durch Rechtsklick und Auswahl des Befehls **Input/Output/Fault-Element löschen** gelöscht werden. Im Fall der Operation `EchoString` benennen Sie die Input und Output Message in `EchoStringRequest` bzw. `EchoStringResponse` um.



Hinzufügen von Parametern zu Messages
Jede Input oder Output Message wird mit einem einzigen Default Message Part (oder Parameter) vom Typ `xs:string` erstellt (*siehe Abbildung unten*). Um einen weiteren Parameter hinzuzufügen, klicken Sie mit der rechten Maustaste entweder auf die Message oder auf einen ihrer Parameter und wählen Sie den Befehl **Message Part (Parameter) hinzufügen**.



Um einen Parameter zu bearbeiten, (i) doppelklicken Sie auf den Text und bearbeiten Sie ihn; (ii) rechtsklicken Sie auf den Parameter und wählen Sie "Bearbeiten" oder (iii) verwenden Sie dien Eingabehilfe "Details" (*siehe Abbildung unten*).

© 2007 Altova GmbH

2.3.3 Erstellen eines Binding

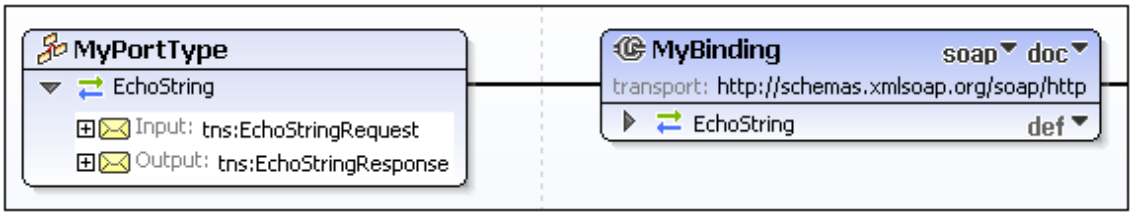
Ein Binding ist ein konkretes Protokoll und eine Datenformatspezifikation für einen bestimmten PortType. Bei Erstellung eines Binding sind daher die folgenden Schritte erforderlich:

- Verküpfung eines Binding mit einem PortType.
- Definition des Binding-Protokolls und der Datenformatsspezifikation.
- Verknüpfung des Binding mit einem Port.

Verknüpfung des Binding mit einem PortType

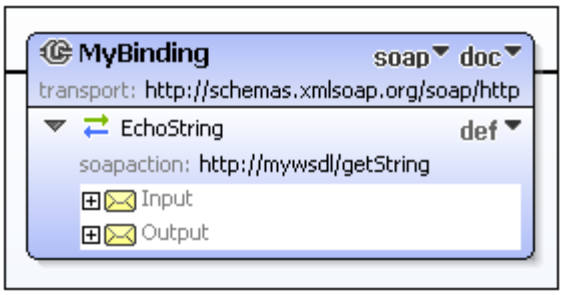
Bei Erstellung eines neuen Binding (rechtsklicken Sie dazu an eine beliebige Stelle in einem leeren Design-Bereich und wählen Sie den Befehl **Binding einfügen**) ist kein PortType mit diesem Binding verknüpft (*Abbildung unten*).

Um einen PortType mit einem Binding zu verknüpfen, klicken Sie auf den Nach-unten-Pfeil des PortType-Eintrags. Daraufhin wird eine Liste der in dem Dokument definierten PortTypes angezeigt. Wählen Sie den PortType, mit dem das Binding verknüpft werden soll. Wenn ein PortType mit dem Binding verknüpft wurde, wird die Verknüpfung folgendermaßen angezeigt:



Auswahl des Protokolls und Datenformats

Das Protokoll des Binding wird durch Klicken auf den Nach-unten-Pfeil des Binding-Felds in der Titelleiste (neben dem Eintrag `soap/http`) und Auswahl eines der drei verfügbaren Protokolle ausgewählt: `SOAP`, `HTTP-GET` und `HTTP-POST` (*Abbildung unten*). Wenn Sie das SOAP Protokoll auswählen, können Sie als Datenformat `document` oder `rpc` auswählen (Diese Listenoptionen erscheinen bei Klick auf den Drowdown-Pfeil rechts von der Protokollauswahl).



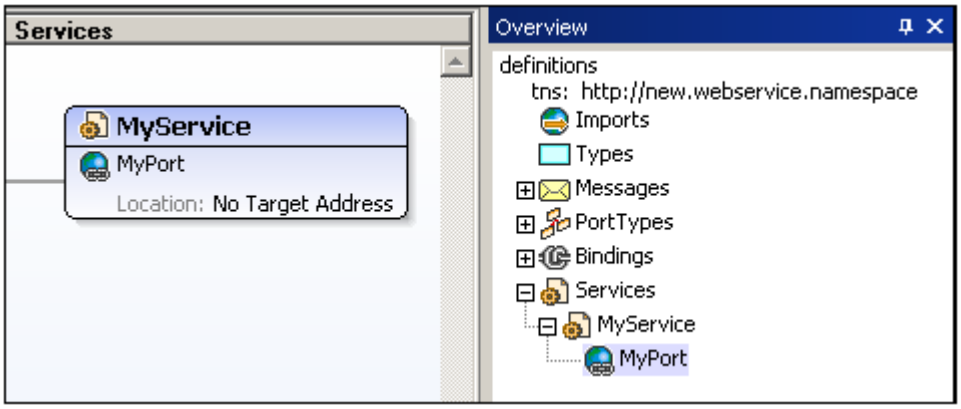
Die `soapAction` für die einzelnen Operationen im Binding können im Design (*siehe Abbildung oben*) oder in der Eingabehilfe "Details" definiert werden, wenn diese Operation ausgewählt ist.

Verknüpfung des Binding mit einem Port

Um das Binding mit einem Port zu verknüpfen, muss zuerst der Port definiert werden. Eine Beschreibung zur Erstellung eines Port in einem Service und Verknüpfen des Port mit einem Binding finden Sie im Abschnitt Erstellen eines Service und von Ports.

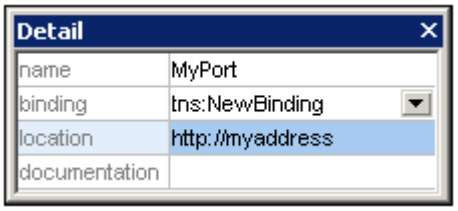
2.3.4 Erstellen eines Service und von Ports

Um einen neuen Service hinzuzufügen, rechtsklicken Sie auf eine leere Stelle im Design und wählen Sie im Kontextmenü den Befehl "Service einfügen". Wenn Sie ein neues WSDL-Dokument angelegt haben, enthält das Design bereits einen Service. Sie können den Service umbenennen, indem Sie auf seinen Namen klicken, ihn bearbeiten und anschließend die **Eingabtaste** drücken. Beachten Sie, dass der Name des Service sich auch in der Eingabehilfe "Übersicht" ändert (*Abbildung unten*).



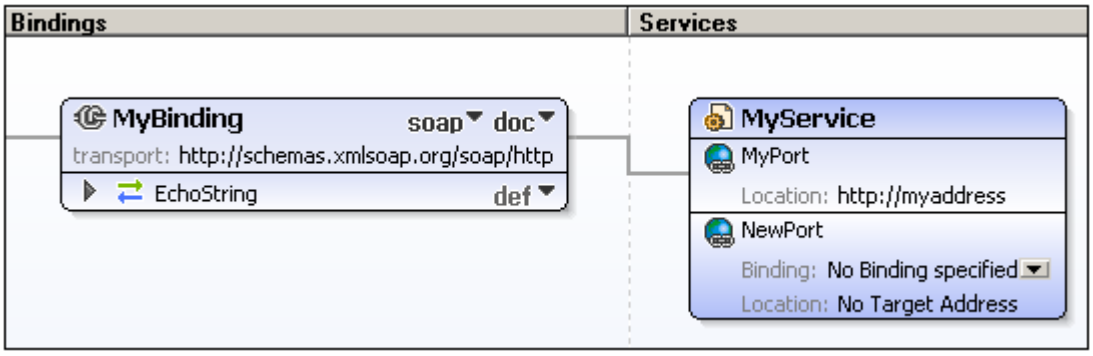
Doppelklicken Sie in der Eingabehilfe "Übersicht" auf den Eintrag `NewPort`, ändern Sie ihn in `MyPort`. und drücken Sie die **Eingabetaste**. Beachten Sie, dass sich der Name des Port auch im Design im Feld `MyService` ändert (*Abbildung oben*). Um zusätzliche Ports hinzuzufügen, rechtsklicken Sie entweder auf den Service oder den Port und wählen Sie im Kontextmenü den Befehl **Port einfügen**.

Die Adresse des Port kann folgendermaßen eingegeben werden: (i) entweder direkt im Design als Wert des Eintrags `Location` (*Abbildung oben*) oder (ii) in der Eingabehilfe "Detail" (durch Doppelklick auf das Feld `Location` und Eingabe der Adresse (*Abbildung unten*).



Verknüpfen eines Binding mit einem Port
Ein Port ist der Endpunkt, der ein Binding mit einer Netzwerk-Adresse kombiniert. Sobald die Adresse der Port definiert wurde, müssen Sie nur noch ein Binding mit dem Port verknüpfen. Klicken Sie dazu auf den Nach-unten-Pfeil für den Binding-Eintrag eines Port und wählen Sie eines aus der Liste der für dieses Dokument definierten Bindings aus.

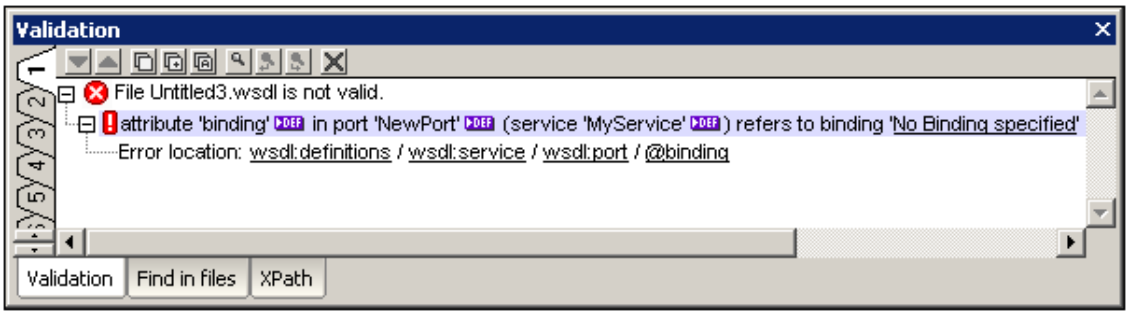
© 2007 Altova GmbH



Anmerkung: Wenn ein Binding bereits mit einem Port verknüpft wurde und Sie ein anderes Binding verknüpfen möchten, müssen Sie die Referenz des Binding entfernen (über das Kontextmenü des Port) und anschließend die neue Binding-Referenz einfügen.

2.3.5 Validieren des WSDL-Dokuments

Nach Fertigstellung des WSDL-Dokuments können Sie es durch Auswahl des Befehls **XML | XML validieren (F8)** validieren. Die Ergebnisse der Validierung werden unterhalb des Hauptfensters in der Validierungsausgabeleiste angezeigt (*Abbildung unten*).



Es werden detaillierte Informationen über jeden gefundenen Fehler angezeigt, sodass Sie diesen schnell finden und beheben können.

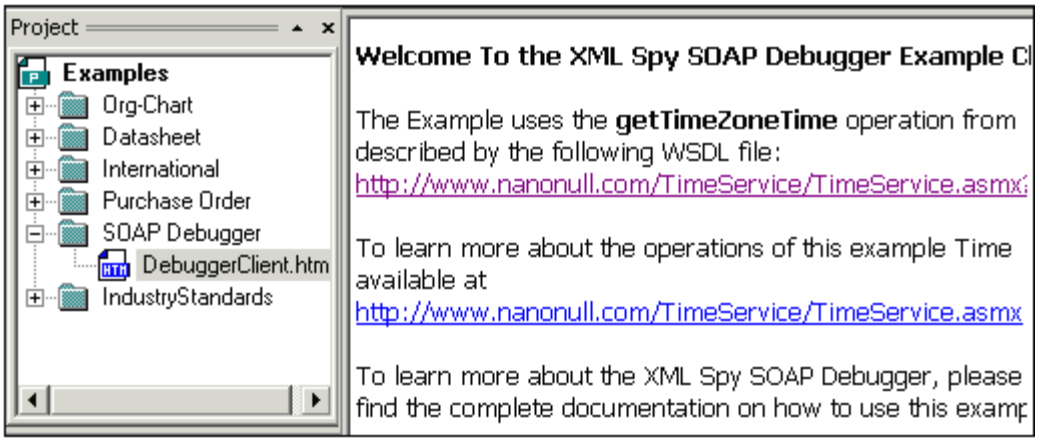
2.3.6 Herstellen einer Verbindung zu einem Webservice und Öffnen von Dateien

In diesem Abschnitt wird Folgendes beschrieben:

- Zugriff auf einen Webservice mittels XMLSpy
- Öffnen einer WSDL-Datei im Web in XMLSpy
- Öffnen eines mit dem WSDL-Dokument verknüpften XML-Schemas in XMLSpy

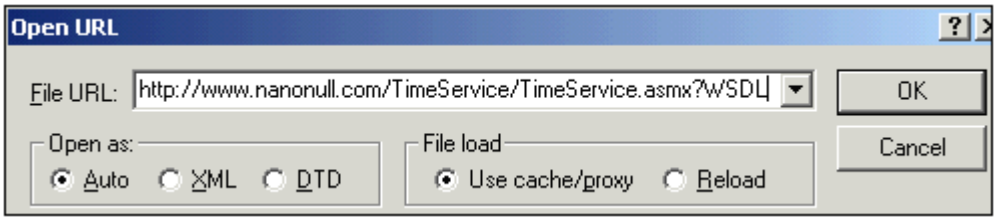
Zugriff auf einen Webservice
Ein Webservice wird normalerweise über eine HTML-Seite wie z.B. `DebuggerClient.htm` aufgerufen. Im Projekt `Examples` finden Sie einen Link dazu. Um diesen Webservice aufzurufen, gehen Sie folgendermaßen vor:

1. Aktivieren Sie das Projektfenster, fallls es nicht zu sehen ist (über die Menüoption **Fenster | Projekt-Fenster**).
2. Klicken Sie auf das Erweiterungssymbol neben dem Ordner "SOAP Debugger" und doppelklicken Sie auf die Datei `DebuggerClient.htm`. Daraufhin wird im Hauptfenster der SOAP Debugger Beispiel Client geöffnet.



Öffnen einer WSDL-Datei in XMLSpy
Um eine webbasierte WSDL-Datei in XMLSpy zu öffnen, gehen Sie folgendermaßen vor:

1. Wählen Sie die Menüoption **Datei | URL öffnen** und geben Sie in im Dialogfeld "URL öffnen" im Feld "Datei-URL" die Adresse `http://www.nanonull.com/TimeService/TimeService.asmx?WSDL` ein oder kopieren Sie sie hinein.



2. Klicken Sie auf **OK,** um die WSDL-Datei zu laden. Die WSDL-Datei wird in der Text-Ansicht angezeigt.
3. Wählen Sie die Menüoption **Datei | Speichern unter** und speichern Sie die Datei lokal z.B. unter dem Namen `timeservice.wsdl`.
4. Klicken Sie auf das WSDL-Ansichtsregister, um die Datei im grafischen WSDL-Editor anzuzeigen.

Anzeigen der mit der aktiven WSDL-Datei verknüpften Schema-Datei
Während im Hauptfenster die Datei `timeservice.wsdl` das aktive Dokument ist, wählen Sie die Menüoption **WSDL | Types | Schema(s) in Schema-Ansicht bearbeiten**. Daraufhin wird die Schema-Datei geöffnet, in der alle in der Datei `timeservice.wsdl` verwendeten Datentypen definiert sind. Sie können das Schema ändern und die Änderungen speichern. Diese Änderungen werden erst dann in der WSDL-Datei übernommen, sobald die Datei erneut geparst wird.

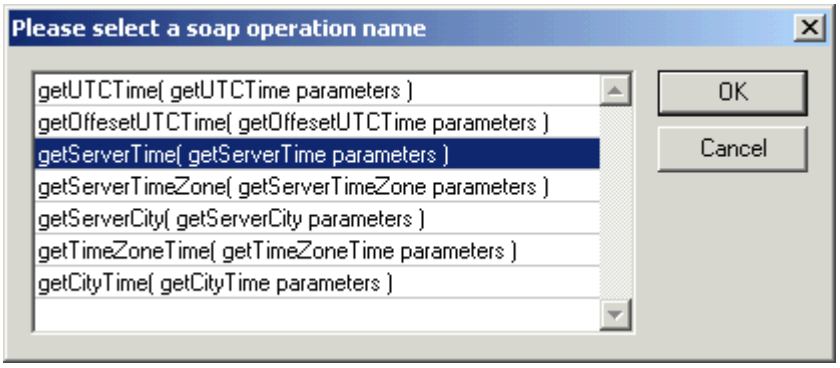
2.3.7 Senden eines SOAP Request von der WSDL-Datei

Um einen SOAP Request von der Datei `timeservice.wsdl` zu senden, gehen Sie folgendermaßen vor:

1. Machen Sie `timeservice.wsdl` im Hauptfenster zur aktiven Datei.
2. Wählen Sie die Menüoption **SOAP | Neuen SOAP Request erstellen**.
3. Suchen Sie die Datei `timeservice.wsdl` und bestätigen Sie die Auswahl mit **OK**.

 © 2007 Altova GmbH

4. Wählen Sie im Dialogfeld, das daraufhin angezeigt wird (*Abbildung unten*), eine SOAP-Operation, z.B. `getServerTime` und klicken Sie auf **OK**.



Daraufhin wird ein neues SOAP Request-Dokument erstellt, das die Operation `getServerTime` enthält.

5. Wählen Sie die Menüoption **SOAP | Request an Server senden**. Das SOAP Response-Dokument wird im Hauptfenster angezeigt. Es enthält das Element `getServerTimeResult`, welches die aktuelle Server-Uhrzeit des Nanonull.com Time Service anzeigt.

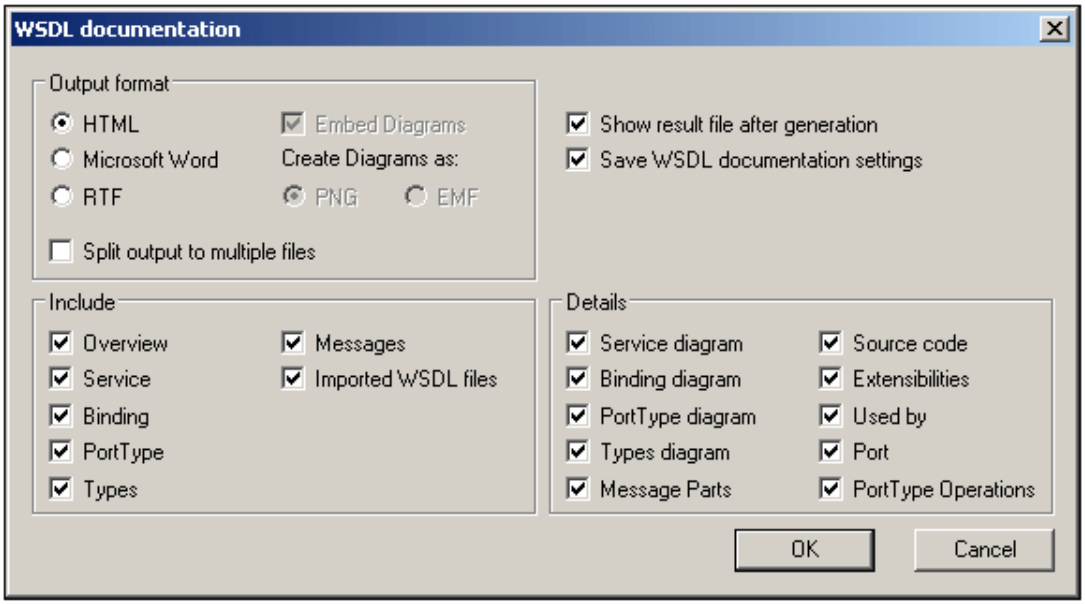
2.3.8 Erstellen von WSDL-Dokumentation

Mit Hilfe der Option **WSDL | Dokumentation generieren** können Sie eine detaillierte Dokumentation des aktuellen WSDL-Dokuments generieren. Sie können die Dokumentation als MS Word-, RTF- oder HTML-Datei ausgeben und angeben, welche Komponenten inkludiert werden sollen. Miteinander in Zusammenhang stehende WSDL-Elemente werden in der generierten Dokumentation durch Hyperlinks verbunden, um die Navigation zu vereinfachen.

Anmerkung: Um Dokumentation im MS Word-Format generieren zu können, muss MS Word (Version 2000 oder höher) installiert sein.

Um Dokumentation für die WSDL-Datei zu generieren, gehen Sie folgendermaßen vor:

1. Machen Sie `timeservice.wsdl` zum aktiven Dokument.
2. Wechseln Sie in die **Schema/WSDL-**Ansicht.
3. Wählen Sie die Menüoption **WSDL | Schema-Dokumentation generieren**. Daraufhin wird das Dialogfeld "WSDL-Dokumentation" (*siehe Abbildung unten*)geöffnet.
4. Wählen Sie den gewünschten Ausgabetyp: **HTML, MS Word** oder **RTF** und
4. Wählen Sie die spezifischen WSDL-Komponenten, die Sie in der Dokumentation berücksichtigen möchten und legen Sie weitere Optionen fest (siehe WSDL-Dokumentationsoptionen unten).



5. Klicken Sie auf OK und geben Sie den Namen der WSDL-Dokumentationsdatei im Dialogfeld "Speichern unter" ein.

WSDL-Dokumentationsoptionen
Sie können zwischen den folgenden Dokumentationsoptionen wählen:

- Das gewünschte Format wird im Bereich "Output-Format" definiert: entweder HTML, MS Word oder RTF. Die Dokumentation kann entweder als eine einzige Datei generiert oder in mehrere Dateien aufgeteilt werden. Wenn mehrere Dateien generiert werden, entspricht jede Datei einer Komponente. Welche Komponenten in der Ausgabe enthalten sein sollen, wird mittels der Kotrollkästchen im Bereich "Inkludieren" festgelegt.
- Die Option "Diagramme einbetten" ist für die Ausgabeoptionen MS Word und RTF aktiv. Wenn diese Option aktiviert ist, werden Diagramme entweder im PNG- oder im EMF-Format in die Ergebnisdatei eingebettet. Andernfalls werden die Diagramme als PNG- oder EMF-Dateien generiert und in der Ergebnisdatei über Objektlinks angezeigt. Wenn es sich bei der Ausgabedatei um eine HTML-Datei handelt, werden alle Diagramme als dokumentexterne PNG-Dateien generiert.
- Im Bereich **Inkludieren** können Sie auswählen, welche Datenelemente in der Dokumentation inkludiert werden sollen. Bei Auswahl der Option "Index" werden alle verwandten Schemas mit ihren globalen Komponenten nach Komponententyp geordnet am Anfang der Datei aufgelistet.
- Im Bereich **Details** werden die Einzelheiten aufgelistet, die für die einzelnen Komponenten inkludiert werden können. Wählen Sie die gewünschten Informationen aus.
- Die Option "Ergebnisdatei nach der Generierung anzeigen" ist für alle drei Ausgabeoptionen aktiv. Wenn diese Option aktiviert ist, wird die erzeugte Datei in der Browser-Ansicht (HTML-Ausgabe), MS Word (MS Word Ausgabe) und der Standardapplikation für .rtf-Dateien (RTF-Ausgabe) angezeigt.

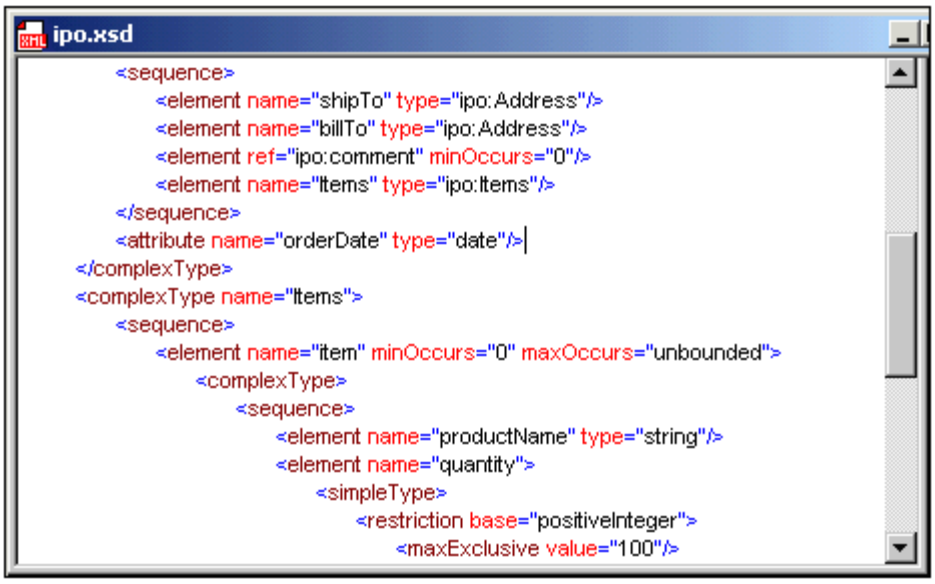
© 2007 Altova GmbH

3 Textansicht

In der Textansicht können Sie Ihren Text - Markup und Content - direkt eingeben. Die Textansicht bietet eine Reihe von Funktionen, um Ihr Dokument schnell und korrekt zu erstellen. Zu den wichtigsten Funktionen zählen die folgenden:

- Visuelle Funktionen, zur Verbesserung der Lesbarkeit des Dokuments. Dazu gehören anpassbare Syntaxfärbung (u.a. die Möglichkeit, serverseitigen VBScript- oder JScript-Code in ASP-Seiten zu markieren), Zeilennummerierung, Lesezeichen, ein- und ausklappbare Elemente, Einzüge, anpassbare Schriftarten und Zeilenumbrüche.
- Intelligente Editierfunktionen wie z.B. die Autokomplettierung von Tags und die automatische Eingabe von Attributen und Child-Elementen.
- Kontextsensitive Eingabehilfen, die Ihnen die Elemente, Attribute und Entities, die an der Cursorposition eingefügt werden können, zur Verfügung stellen.
- Drag-and-drop-Funktionen und Funktionen zum Kopieren und Einfügen.

Diese Funktionen werden in den folgenden Kapiteln dieses Abschnitts näher beschrieben.



Um die Textansicht eines Dokuments zu öffnen, klicken Sie auf die Schaltfläche "Text" am unteren Rand des Dokumentenfensters oder wählen Sie **Ansicht | Text**. Die Textansicht dient zum Bearbeiten von Textdateien aller Art, einschließlich nicht-XML-Dokumenten.

3.1 Visuelle Editierhilfen in der Textansicht

Die Textansicht enthält Funktionen, die die Bearbeitung großer Textpassagen erleichtern: Zeilennummerierung, Lesezeichen, Klappleiste und Einrücklinien. Jede dieser Funktionen kann durch Klicken auf das entsprechende Symbol in der Symbolleiste aktiviert bzw. deaktiviert werden.

Aktiviert/deaktiviert die Zeilennummerierung in allen offenen Dokumenten

Aktiviert/deaktiviert die Klappleiste in allen offenen Dokumenten

Aktiviert/deaktiviert die Lesezeichenleiste

Dient zum Einfügen/Löschen von Lesezeichen

Aktiviert/deaktiviert die Einrücklinien

In der unten gezeigten Abbildung sehen Sie diese Funktionen:

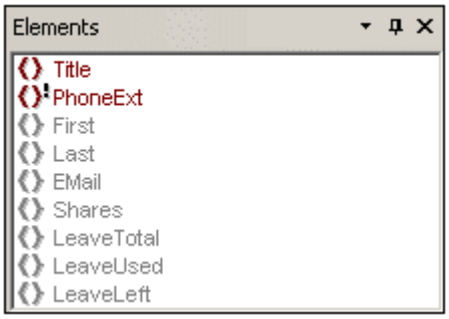
Beachten Sie die folgenden Funktionen in der oben stehenden Abbildung:

- Die Zeilennummerierung ist aktiviert
- Die Klappleiste ist aktiviert. Mit dem Plus-Zeichen werden eingeklappte (reduzierte) Elemente gekennzeichnet. Mit dem Minus-Zeichen werden ausgeklappte (erweiterte) Elemente gekennzeichnet. Wenn ein Element eingeklappt ist, wird auch die Zeilennummerierung eingeklappt (siehe eingeklappte Zeilen 14 und 24). Wird die Klappleiste deaktiviert, werden alle eingeklappten Elemente automatisch ausgeklappt.
- Die Lesezeichenleiste ist aktiviert. In Zeile 11 wurde ein Lesezeichen gesetzt.
- Die Einrücklinien sind aktiviert. Dabei handelt es sich um vertikale gepunktete Linien, anhand derer Sie besser sehen, wo ein Element beginnt und wo es endet (siehe Zeilen 46 und 47).

© 2007 Altova GmbH

3.2 Eingabehilfen in der Textansicht

Element-Eingabehilfe
In der Textansicht werden Elemente, die an der Cursorposition eingefügt werden können, in der Element-Eingabehilfe dunkelrot angezeigt. Obligatorische Elemente haben ein Ausrufezeichen vor dem Elementnamen. Geschwisterelemente von zulässigen Elementen, die selbst an der Cursorposition nicht eingefügt werden können, werden grau angezeigt. Wenn sich die Position des Cursors ändert, ändert sich auch die Liste in der Eingabehilfe und zeigt nur diejenigen Elemente in Rot an, die an dieser Stelle eingefügt werden können, während andere Geschwisterelemente grau angezeigt werden.



Um ein Element an der Cursorposition einzufügen, doppelklicken Sie auf das gewünschte Element. Daraufhin werden der Start- und End-Tag des Elements eingefügt. Auch obligatorische Elemente werden eingefügt, wenn diese Option im Dialogfeld **Optionen** (**Extras | Optionen | Bearbeiten**) aktiviert wurde.

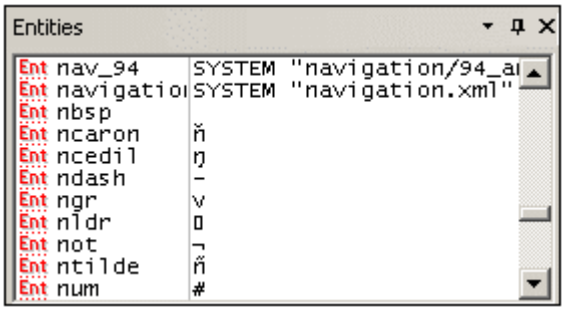
Hinweis: Im Dialogfeld **Optionen** (**Extras | Optionen | Bearbeiten**) können Sie festlegen, dass beim Einfügen eines Elements auch obligatorische Child-Elemente eingefügt werden können.

Attribut-Eingabehilfe
Wenn Sie den Cursor in der Textansicht innerhalb des Start-Tags eines Elements und hinter einem Leerzeichen positionieren, werden die für dieses Element deklarierten Attribute angezeigt. Nicht verwendete Attribute werden rot angezeigt, verwendete grau. Obligatorische Attribute weisen ein Ausrufezeichen "**!**" vor dem Namen des Attributs auf.

Um ein Attribut einzufügen, doppelklicken Sie auf das gewünschte Attribut. Das Attribut wird nun an der Cursorposition zusammen mit einem Ist-gleich-Zeichen und Anführungszeichen, innerhalb derer der Attributwert gesetzt wird, angezeigt. Der Cursor wird zwischen die Anführungszeichen gesetzt, sodass Sie den Attributwert direkt eingeben können.

Hinweis: Bestehende Attribute, die kein zweites Mal im aktuellen Element eingefügt werden dürfen, erscheinen grau.

Entities-Eingabehilfe
Alle geparsten oder ungeparsten Entities, die innerhalb einer Zeile (innerhalb des XML-Dokuments) oder in einer externen DTD deklariert sind, werden in der Entities-Eingabehilfe angezeigt.



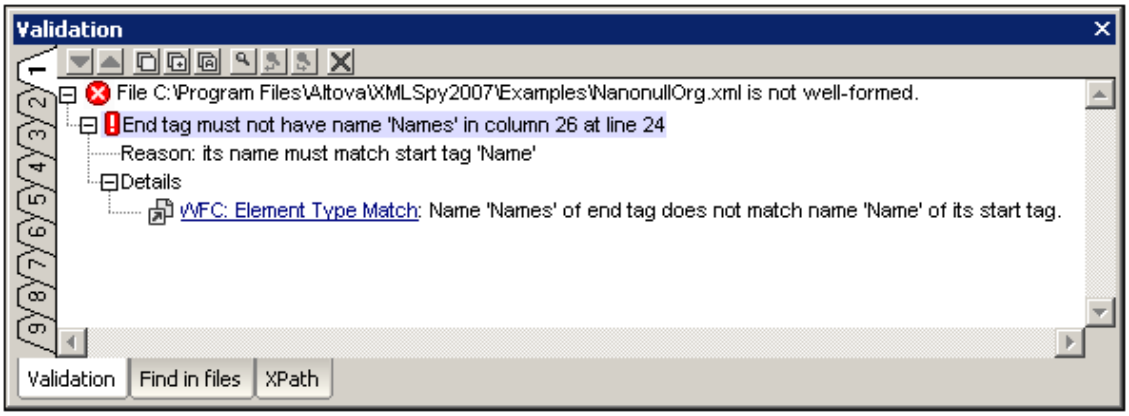
Um eine Entity an der Cursorposition einzufügen, doppelklicken Sie auf die gewünschte Entity.

Hinweis: Wenn Sie eine interne Entity einfügen, müssen Sie Ihr Dokument speichern und erneut öffnen, damit die Entity in der Entities-Eingabehilfe angezeigt wird.

© 2007 Altova GmbH

3.3 Ausgabefenster

Die Ausgabefenster (*Abbildung unten*) befinden sich unterhalb des Hauptfensters. Beim Öffnen von XMLSpy werden sie automatisch eingeblendet. Um die Ausgabefenster auszublenden, bzw. wieder einzublenden, klicken Sie auf **Fenster | Ausgabefenster**.



Es gibt drei Ausgabefenster, die Sie jeweils durch Klicken auf das entsprechende Register aktivieren können:

- Validierung: Hier werden die Ergebnisse der Validierung von XML-, XML-Schema, DTD- oder XQuery-Dokumenten angezeigt.
- In Dateien suchen: Hier werden die Ergebnisse von in mehreren Dateien durchgeführten Such- und Ersetzungsvorgängen angezeigt.
- XPath-Auswertung: Hier werden die Ergebnisse angezeigt, die durch Auswertung eines XPath-Ausdrucks gegen das aktive XML-Dokument erzeugt werden.

Die oben angeführten Operationen werden in den Unterabschnitten dieses Abschnitts näher beschrieben.

Allen Ausgabefenstern gemeinsame Features

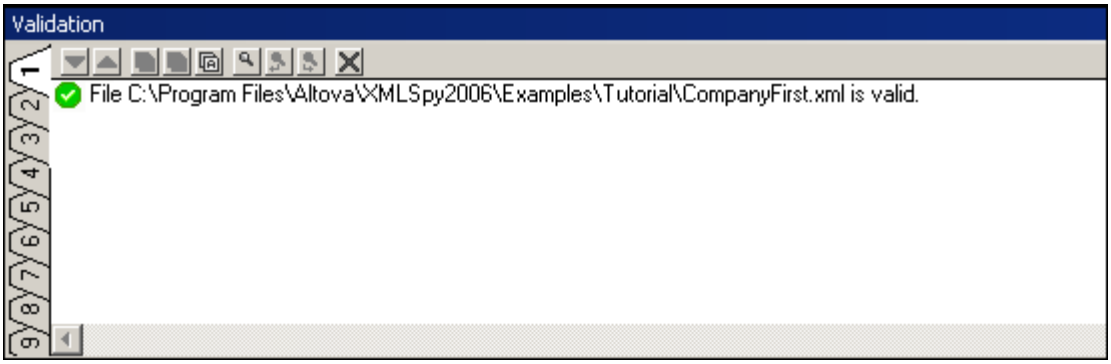
Jedes Ausgabefenster hat neun Register. Um ein Register zu aktivieren, müssen Sie darauf klicken. Das Ergebnis der aktuellen Validierung, des aktuellen Ersetzungsvorgangs oder der aktuellen XPath-Auswertung wird auf dem aktiven Register angezeigt. Wenn ein Ergebnis bereits auf einem Ausgaberegister angezeigt wird, wird es durch das neue Ergebnis ersetzt. Der Vorteil von mehreren Registern ist, dass das Ergebnis auf einem Register zur Verfügung steht, (i) und mit einem neuen Ergebnis auf einem anderen Register verglichen werden kann (ii) oder zu einem späteren Zeitpunkt wieder herangezogen werden kann.

Jedes Ausgabefenster enthält am oberen Rand eine Gruppe von Symbolen (*Abbildung unten*).

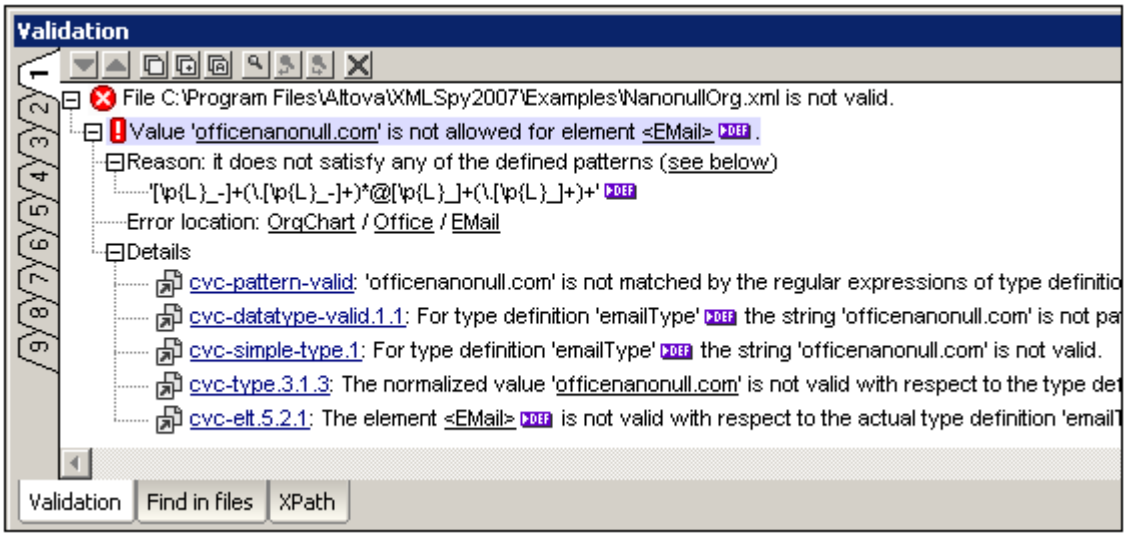
Die ersten beiden Symbole von links dienen zum nach oben und unten Navigieren in den Ausgabemeldungen. Die nächsten drei Symbole dienen zum Kopieren von Meldungen in die Zwischenablage. Sie können eine Meldung durch Anklicken auswählen. Die nächsten drei Symbole sind auf dem Register "In Dateien suchen" aktiv und dienen zum Navigieren in den gefundenen Einträgen. Das Symbol ganz rechts dient zum Löschen des aktuellen Ausgaberegisters.

3.3.1 Validierung

Wenn ein XML-, XML-Schema, DTD- oder XQuery-Dokument gültig ist, wird im Validierungsfenster eine entsprechende Meldung (*Abbildung unten*) angezeigt.



Andernfalls wird eine Meldung, die den Fehler beschreibt, angezeigt (*Abbildung unten*). Beachten Sie, dass es Links (schwarzen Text) zu Nodes und Node-Inhalt im XML-Dokument gibt sowie Links zum entsprechenden Abschnitt in der relevanten Spezifikation, in dem die betreffende Regel beschrieben wird (blauer Text). Wenn Sie auf eines der violetten Dev-Symbole klicken, wird die entsprechende Schemadefinition in der Schema-Ansicht geöffnet.



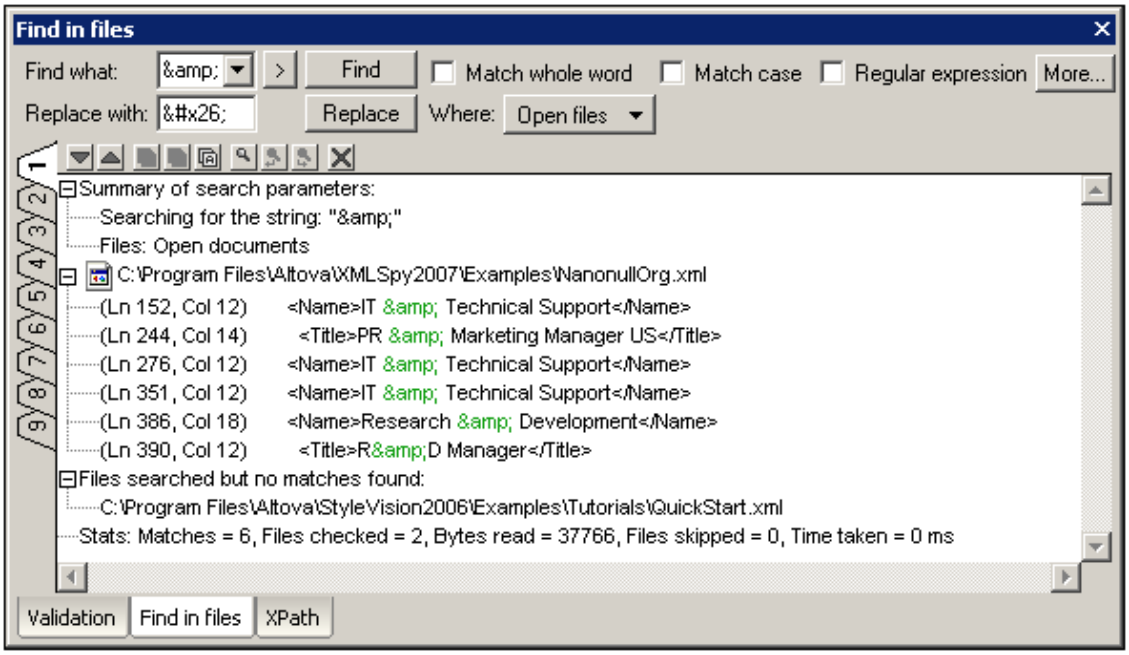
Das Validierungs-Ausgabefenster ist in allen Ansichten aktiv. Wenn Sie jedoch auf einen Link zum XML-Dokumentinhalt klicken, wird der entsprechende Node in der Text-Ansicht markiert. Beachten Sie jedoch: Wenn ein XML-Schema in der Schema-Ansicht validiert wird, ändert sich die Ansicht bei Klicken auf das Dev-Symbol nicht.

Anmerkung: Der Befehl **Validieren** (im Menü XML) wird normalerweise auf das aktive Dokument angewendet. Sie können den Befehl jedoch auch auf eine Datei oder eine Gruppe von Dateien im aktiven Projekt anwenden. Wählen Sie die gewünschte Datei bzw. den gewünschten Ordner im Projekt-Fenster (durch Anklicken) aus und klicken Sie auf **XML | Validieren** oder **F8**. Ungültige Dateien in einem Projekt werden geöffnet und im Hauptfenster aktiviert. Falls die Datei ungültig ist, wird eine entsprechende Fehlermeldung angezeigt.

© 2007 Altova GmbH

3.3.2 In Dateien suchen

Im Ausgabefenster "In Dateien suchen" (*Abbildung unten*) können Sie Such- und Ersetzungsvorgänge schnell in mehreren Dokumenten gleichzeitig durchführen. Es bietet Funktionen zum raschen Navigieren in den gefundenen Instanzen. Die Ergebnisse der einzelnen Such- und Ersetzungsvorgänge werden auf einem der von 1 bis 9 durchnummerierten Register angezeigt. Wenn Sie in der Ergebnisliste auf eine der gefundenen Instanzen klicken, gelangen Sie in der Text-Ansicht des Dokuments zu dieser Instanz.



Suchkriterien
Es gibt zwei grundlegende Suchkriterien: (i) Was soll gesucht werden und (ii) wo soll gesucht werden?

Was soll gesucht werden: Der zu suchende String wird in das Textfeld "Suchen nach" eingegeben. Wenn mit diesem String ein ganzes Word gefunden werden soll, muss das Kontrollkästchen "Ganzes Wort" aktiviert werden. Um z.B. den String `fit` zu finden wird bei aktivierten Kontrollkästchen "Ganzes Wort" nur das Wort `fit` gefunden; `fit` in `fitness` würde z.B. nicht gefunden. Mit Hilfe des Kontrollkästchens "GROSS/klein beachten" können Sie angeben, ob die Groß- und Kleinschreibung beachtet werden soll. Wenn es sich bei dem im Textfeld "Suchen nach" eingegebenen Text um eine Regular Expression handelt, muss das Kontrollkästchen "Regular Expression" aktiviert werden. Um eine Eingabehilfe zur Eingabe von Regular Expressions aufzurufen, klicken Sie auf die Schaltfläche [>]. Die Verwendung von Regular Expressions zur Suche wird unter Suchen erklärt. Über die Schaltfläche **Mehr** wird das Dialogfeld In Dateien suchen geöffnet. Hier können Sie weitere Suchbedingungen und Aktionen definieren. Nähere Informationen finden Sie unter Bearbeiten | In Dateien suchen.

Wo soll gesucht werden: Die Suche kann in folgenden Dateien durchgeführt werden: (i) in allen auf der Benutzeroberfläche geöffneten Dateien; (ii) in den Dateien des aktuellen Projekts; und (iii) in den Dateien eines ausgewählten Ordners. Im Dialogfeld "In Dateien suchen" (das durch Klicken auf **Mehr** aufgerufen wird) können Sie zusätzliche Bedingungen festlegen.

Ersetzen durch
Der String, durch den der gefundene String ersetzt werden soll, wird in das Textfeld "Ersetzen durch" eingegegeben. Beachten Sie: Wenn das Textfeld "Ersetzen durch" leer ist und Sie auf die Schaltfläche "Ersetzen" klicken, wird der gefundene Text duch einen leeren String ersetzt.

Die Ergebnisse
Nachdem Sie auf die Such- oder Ersetzungsschaltflächen geklickt haben, werden die Ergebnisse des Such- bzw. Ersetzungsvorgangs im Ausgabefenster "In Dateien suchen" angezeigt. Die Ergebnisse sind in vier Teile untergliedert:

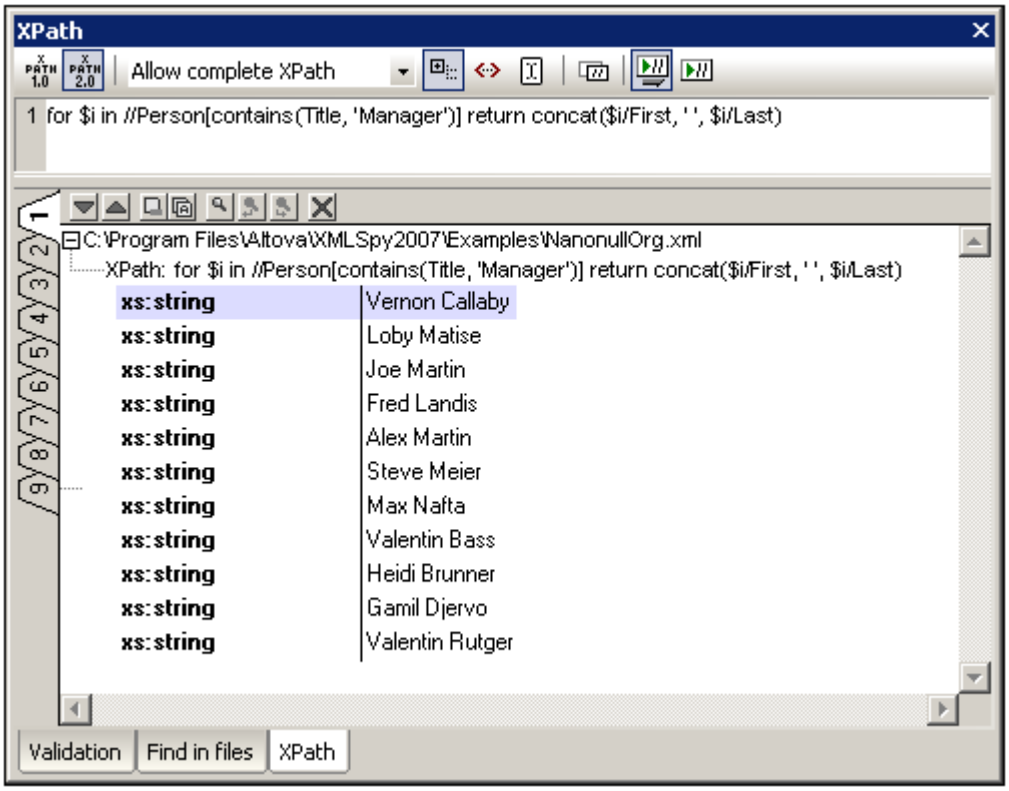
- Eine Zusammenfassung der Suchparameter, in der aufgelistet ist, welcher Suchstring gesucht wurde und welche Dateien durchsucht wurden.
- Eine Liste der gefundenen bzw. ersetzten Strings (je nachdem, ob Sie die Schaltfläche **Suchen** oder **Ersetzen** betätigt haben). Die Einträge in dieser Liste sind Links zum gefundenen/ersetzten Text in der Text-Ansicht des Dokuments. Wenn das Dokument nicht geöffnet ist, wird es in der Text-Ansicht geöffnet und der gefundene/ersetzte Text wird markiert.
- Eine Liste der Dateien, die durchsucht wurden, in denen jedoch keine Instanzen gefunden wurden.
- Eine Zusammenfassung der Statistiken für die Suchaktion einschließlich der Anzahl der Treffer und der überprüften Dateien.

Anmerkung: Beachten Sie: Mit der Funktion "In Dateien suchen" werden die Befehle zum Suchen und Ersetzen gleichzeitig an mehreren Dateien durchgeführt. Die Ergebnisse werden im Ausgabefenster "In Dateien suchen" angezeigt. Um eine Suche durchzuführen, bei der Sie von einer gefundenen Instanz zur nächsten springen, verwenden Sie den Befehl Suchen.

3.3.3 XPath

Im Ausgabefenster für die XPath-Auswertung können Sie bis zu neun XPath-Ausdrücke auswerten, jeden davon im Zusammenhang mit dem gerade aktiven Dokument D.h. Sie können auf jedem der neun Ausgaberegister des XPath-Auswertungsfensters einen anderen XPath-Ausdruck haben. Wenn im Hauptfenster ein XML-Dokument aktiv ist, kann ein XPath-Ausdruck (d.h. ein Register) ausgewertet werden und der Ausdruck kann am gerade aktiven Dokument ausgeführt werden. Auf diese Art können Sie vergleichen, wie sich die verschiedenen XPath-Ausdrücke (auf den einzelnen Registern) auf das gerade aktive Dokument auswirken. Zusätzlich dazu können Sie einen (oder mehrere) XPath-Ausdrücke an mehreren XML-Dokumenten auswerten; machen Sie dazu ein anderes Dokument zum aktiven und klicken Sie auf das Symbol "XPath-Ausdruck auswerten".

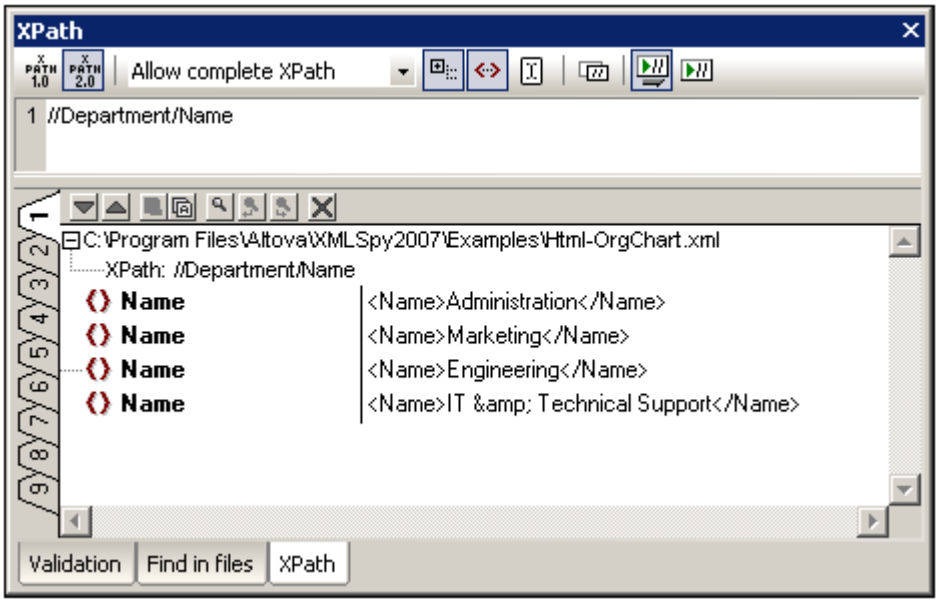
© 2007 Altova GmbH



Eine großer Vorteil des Fensters "XPath" ist, dass Sie den XPath-Ausdruck, das Ergebnis der Auswertung sowie das XML-Dokument, auf das der XPath-Ausdruck angewendet wird, in einer einzigen Ansicht sehen. Dadurch können Sie (i) im XML-Dokument navigieren, während Sie den XPath-Ausdruck und die Ergebnisse sehen und (ii) den XPath-Ausdruck bei eingeblendetem XML-Dokument sehen.

Schritte bei der Auswertung eines XPath-Ausdrucks
Im Folgenden sehen Sie die wichtigsten Schritte bei der Verwendung des Fensters "XPath".

1. Wählen Sie die gewünschte XPath-Version aus (`1.0` oder `2.0`), indem Sie auf das entsprechende Symbol in der Symbolleiste des Ausgabefensters klicken.
2. Wählen Sie die Art des XPath-Ausdrucks aus der Dropdown-Auswahlliste aus. Normalerweise benötigen Sie die Option `Vollständigen XPath gestatten`. Die Optionen `XPath-Schema-Selektor` und `XPath-Schema-Feld` können verwendet werden, wenn im XML-Schema eindeutige Identity Constraints definiert wurden.
3. Aktivieren Sie das Symbol "XPath-Ausdruck bei der Eingabe auswerten", wenn der XPath-Ausdruck schon während der Eingabe ausgewertet werden soll. Wenn dieses Symbol deaktiviert ist, wird der Ausdruck erst dann ausgewertet, wenn Sie auf das Symbol "XPath-Ausdruck auswerten" klicken.
4. Aktivieren Sie das Symbol "Kopfzeile in der Ausgabe anzeigen", um den Pfad zur XML-Datei und dem XPath-Ausdruck in der Ausgabe anzuzeigen (*siehe Abbildung unten*).
5. Wenn der XPath-Ausdruck Nodes zurückgibt - wie z.B. Elemente oder Attribute - können Sie auswählen, ob die einzelnen Nodes nach Node-Namen aufgelistet werden sollen oder ob auch der Inhalt des Node angezeigt werden soll. In der unten gezeigten Abbildung wird auch der Inhalt der Nodes angezeigt. (in der Spalte rechts).



Um die Option zur Anzeige des Node-Inhalts auszuwählen, aktivieren Sie das Symbol "Gesamtes Ergebnis anzeigen".

6. Um einen XPath-Ausdruck relativ zu einer Auswahl im XML-Dokument zu definieren, aktivieren Sie das Symbol "Aktuelle Auswahl in Dokument als Ursprung für XPath-Ausdruck festlegen".
7. Geben Sie den XPath-Ausdruck ein. Wenn der Ausdruck mehrere Zeilen lang sein ist, drücken Sie die **Eingabetaste**.
8. Um den Ausdruck auszuwerten, klicken Sie auf das Symbol "XPath-Ausdruck auswerten" (wenn die Option "XPath-Ausdruck bei der Eingabe auswerten" deaktiviert ist oder wenn ein neues XML-Dokument zum aktiven gemacht wird).

Funktionen des Ergebnisfensters
Das Ergebnisfenster hat zwei sehr nützliche Features:

- Wenn das Ergebnis - im Gegensatz zu mit einem Ausdruck generierten Literalen - einen Node enthält (auch einen Text-Node) wird der Node im XML-Dokument im Hauptfenster markiert, wenn Sie im Ergebnisfenster darauf klicken.
- Wenn ein Node im Ergebnisfenster ausgewählt wird, wird bei Auswahl des Symbols "Kopiert den XPath-Ausdruck von der aktuellen Auswahl in das Bearbeiten-Feld" ein XPath-Ausdruck generiert, um den ausgewählten Node zu finden. Beachten Sie, dass der gerade im Bearbeiten-Feld angezeigte XPath-Ausdruck durch den neuen Ausdruck überschrieben wird.

XPath-Ausdrücke im Bearbeiten-Feld
Beachten Sie bitte die folgenden allgemeinen Punkte zu XPath-Ausdrücken. XPath- 1.0 und XPath 2.0-spezifische Punkte werden weiter unten gesondert abgehandelt.

- Wenn Sie im Bearbeiten-Feld einen Ausdruck eingeben, wird er schwarz angezeigt, wenn die Syntax korrekt ist und rot, wenn die Syntax nicht korrekt ist. Beachten Sie, dass eine korrekte Syntax nicht bedeutet, dass der Ausdruck keinen Fehler enthält. Die Fehlermeldungen werden im Ergebnisfenster angezeigt.
- Ein Fehler kann dadurch entstehen, dass XPath-Evaluator Optionen falsch eingestellt wurden. Stellen Sie daher die richtigen Optionen ein, v.a. die XPath-Version und den XPath-Ursprung. (absolut oder relativ zur aktuellen Auswahl im derzeit aktiven XML-Dokument)

© 2007 Altova GmbH

- Nur Namespaces, die sich am Element, an dem der XPath seinen Ursprung hat (d.h. am Kontext-Node) im Geltungsbereich befinden, werden ausgewertet.

XPath 1.0-Ausdrücke

- XPath 1.0-Funktionen müssen ohne Namespace-Präfix eingegeben werden.
- Es werden die vier Node-Tests nach Typ unterstützt: `node()`, `text()`, `comment()` und `processing-instruction()`.

XPath 2.0-Ausdrücke

- Es werden String- (z.B. 'Hello') und numerische Literale (z.B. 256) unterstützt. Zum Erstellen anderer Literale auf Basis vom XML-Schema-Typen wird ein Konstruktor verwenden, dem ein Namespace vorangestellt wird (z.B. `xs:date('2004-09-02')`). Das für XML-Schema-Typen verwendete Namespace-Präfix muss an den XML-Schema-Namespace gebunden werden: `http://www.w3.org/2001/XMLSchema` und dieser Namespace muss in Ihrer XML-Datei deklariert sein.
- Die vom XPath Evaluator verwendeten XPath 2.0-Funktionen gehören zum Namespace `http://www.w3.org/2005/xpath-functions`. Normalerweise ist das Präfix `fn:` an diesen Namespace gebunden. Da es sich bei diesem Namespace jedoch um den vom XPath Evaluator verwendeten Standard-Namespace für Funktionen handelt, müssen Sie bei Funktionen kein Präfix definieren. Wenn Sie ein Präfix verwenden, stellen Sie sicher, dass es an den Namespace für XPath 2.0 Funktionen gebunden ist, den Sie im XML-Dokument deklarieren müssen. Beispiele für die Verwendung einer Funktion: `current-date()` (wobei der Funktions-Namespace im XML-Dokument nicht deklariert ist); `fn:current-date()` (wobei der Funktions-Namespace im XML-Dokument nicht deklariert ist oder im XML-Dokument deklariert ist und an das Präfix `fn:` gebunden ist). Sie können das Namespace-Präfix sogar weglassen, wenn der Funktions-Namespace im XML-Dokument mit oder ohne Präfix deklariert ist; denn eine Funktion, die in einem XPath-Ausdruck so verwendet wird, befindet sich im Standard-Namespace. Dies ist der Standard-Namespace für Funktionen.
- Bei Verwendung der beiden Duration-Datentypen (`yearMonthDuration` und `dayTimeDuration`) muss der XML-Schema-Namespace (`http://www.w3.org/2001/XMLSchema`) in der XML-Datei deklariert werden und das Namepace-Präfix muss bei diesen Typen im XPath-Ausdruck verwendet werden. Beispiel: `years-from-yearMonthDuration(xs:yearMonthDuration('P22Y18M'))`.

Bitte beachten Sie: Zusammenfassend ist zu Namespaces zu sagen: Bei Verwendung von Konstrukoren oder Typen aus dem XML-Schema-Namespace müssen Sie den XML-Schema-Namespace im XML-Dokument deklarieren und im XPath-Ausdruck die korrekten Namespace-Präfixes verwenden. für XPath-Funktionen müssen Sie kein Präfix verwenden.

Datentypen in XPath 2.0
Wenn Sie einen XPath 2.0-Ausdruck für ein XML-Dokument, das ein XML-Schema referenziert und gemäß diesem Schema gültig ist, auswerten, müssen Sie Datentypen, die nicht durch eine Operation implizit in den benötigten Datentyp konvertiert werden, explizit konstruieren oder casten. In dem vom integrierten XPath-Prozessor verwendeten XPath 2.0-Datenmodell werden alle **atomisierten** Node-Werte aus dem XML-Dokument dem Datentyp `xs:untypedAtomic` zugewiesen. Der Typ `xs:untypedAtomic` funktioniert gut bei impliziten Typkonvertierungen. So hat der Ausdruck `xs:untypedAtomic("1") + 1` als Ergebnis den Wert 2, da der

xs:untypedAtomic-Wert implizit durch den Additions-Operator in xs:double umgewandelt wird. Arithmetische Operatoren wandeln Operanden implizit in xs:double um. Vergleichsoperatoren wandeln Operanden vor dem Vergleich in xs:string um.

In einigen Fällen müssen Datentypen jedoch explizit in den benötigten Datentyp konvertiert werden. Wenn Sie z.B. zwei Elemente haben, startDate und endDate, die im XML-Schema als Typ xs:date definiert sind, wird bei Verwendung des XPath 2.0 Ausdrucks endDate - startDate ein Fehler angezeigt. Wenn Sie dagegen xs:date(endDate) - xs:date(startDate) oder (endDate cast as xs:date) - (startDate cast as xs:date) verwenden, wird der Ausdruck korrekt zu einer einzigen Sequenz vom Typ xs:dayTimeDuration ausgewertet.

Bitte beachten Sie: Die XPath-Prozessoren, die vom XPath Evaluator verwendet werden, werden auch vom Altova XSLT-Prozessor verwendet. XPath 2.0 Ausdrücke in XSLT Stylesheets, die nicht implizit in den erforderlichen Datentyp konvertiert werden, müssen explizit als der benötigte Datentyp konstruiert oder gecastet werden.

Stringlänge der Zeichen- und Entity-Referenzen
Bei Verwendung von Zeichen- und Entity-Referenzen als Eingabestring für die Funktion string-length() können die Referenzen nicht aufgelöst werden und es wird die Länge des nicht aufgelösten Textstring zurückgegeben. In einer XSLT-Umgebung hätten diese Referenzen allerdings eine Bedeutung und es wird die Länge des aufgelösten String zurückgegeben.

Fehlerbehebung
Wenn Ihr XPath-Ausdruck einen Fehler zurückgibt, gehen Sie folgendermaßen vor:

1. Überprüfen Sie, ob die Optionen richtig eingestellt wurden (XPath-Version, Ursprung, usw.).
2. Überprüfen Sie, ob die Funktions-, Konstruktor-, Node-Namen usw. korrekt sind.
3. Überprüfen Sie, ob Präfixe erforderlich sind und ob diese Präfixe bei Funktionen, Konstruktoren und Argumenten im XPath-Ausdruck korrekt gesetzt sind.
4. Falls im XML-Dokument Namespaces deklariert sind, überprüfen Sie, ob diese korrekt sind. Die korrekten Namespaces sind:

   XML-Schema: http://www.w3.org/2001/XMLSchema
   XPath 2.0-Funktionen: http://www.w3.org/2005/xpath-functions

Unterstützung von XPath 2.0-Funktionen
Siehe Anhänge.

 © 2007 Altova GmbH

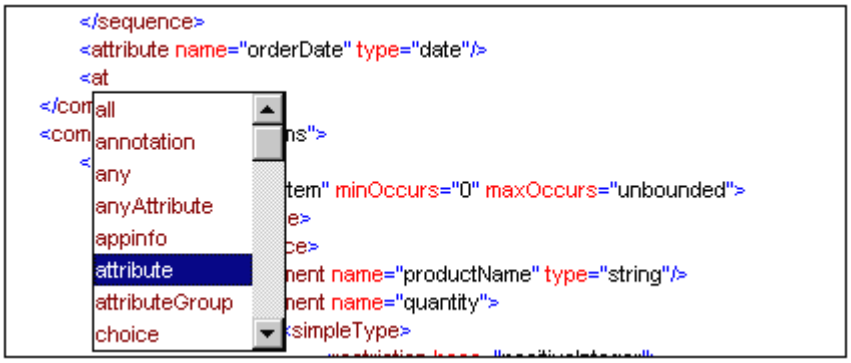
3.4 Bearbeiten von XML-Dokumenten

Syntaxfärbung
Die Syntaxfärbung richtet sich nach der Art des XML Nodes, d.h. je nachdem, ob es sich beim XML Node um ein Element, Attribut, Inhalt, einen CDATA-Abschnitt, Comment oder eine Processing Instruction handelt. Die Texteigenschaften dieser XML Node-Typen können im Dialogfeld "Optionen" (**Extras | Optionen**) auf dem Register "Textschriftarten" eingestellt werden.

Start-Tag und End-Tag
Wenn Sie den Cursor innerhalb des Start- oder End-Tags eines XML-Elements setzen, wird der andere Tag des Paars markiert, wenn Sie **Strg+E** drücken. Bei wiederholtem Drücken von **Strg+E** können Sie zwischen dem Start- und End-Tag hin- und herwechseln. Dies ist ein weiteres Hilfsmittel, um die Start- und End-Tags eines XML-Elements leichter zu finden.

Intelligentes Editieren
Wenn Sie mit einem XML-Dokument arbeiten, das auf einer DTD oder einem XML-Schema basiert, stehen Ihnen in XMLSpy in der Textansicht verschiedene intelligente Editierfunktionen zur Verfügung, mit Hilfe derer Sie schnell das richtige Element, Attribut oder den richtigen Attributwert einfügen können, und zwar dem Content Model entsprechend, das für das Element definiert wurde, das Sie gerade bearbeiten. Intelligentes Editieren funktioniert normalerweise folgendermaßen:

1. Geben Sie **<** (das "kleiner als"-Zeichen) an der Stelle ein, an der Sie ein XML-Element einfügen möchten. Darauf wird eine Popup-Liste angezeigt, die alle Elemente enthält, die an dieser Stelle eingefügt werden dürfen.
2. Geben Sie die ersten paar Zeichen des gewünschten Elements ein. Ein Element in der Popup-Liste, das diese Zeichen enthält, wird markiert.



3. Klicken Sie mit der Maus auf den Eintrag oder drücken Sie die Eingabetaste, um den markierten Eintrag zu übernehmen oder gehen Sie mit Hilfe der Pfeiltasten zum gewünschten Eintrag und klicken Sie darauf oder drücken Sie die Eingabetaste.

Das Popup-Fenster wird auch in den folgenden Fällen angezeigt:

- Wenn Sie die Leertaste drücken, während sich der Cursor zwischen den Tags eines Elements befindet und wenn für dieses Element ein Attribut definiert ist. Das Popup-Fenster enthält alle verfügbaren Attribute.
- Wenn sich der Cursor innerhalb der doppelten Anführungszeichen befindet, die einen Attributwert umschließen, für den eine Aufzählung mit Werten zur Auswahl steht. Im Popup-Fenster werden die aufgezählten Werte angezeigt.
- Wenn Sie das Zeichen **</** (den Beginn eines End-Tags) eingeben, wird im Popup-Fenster der Name des zu schließenden Elements angezeigt.

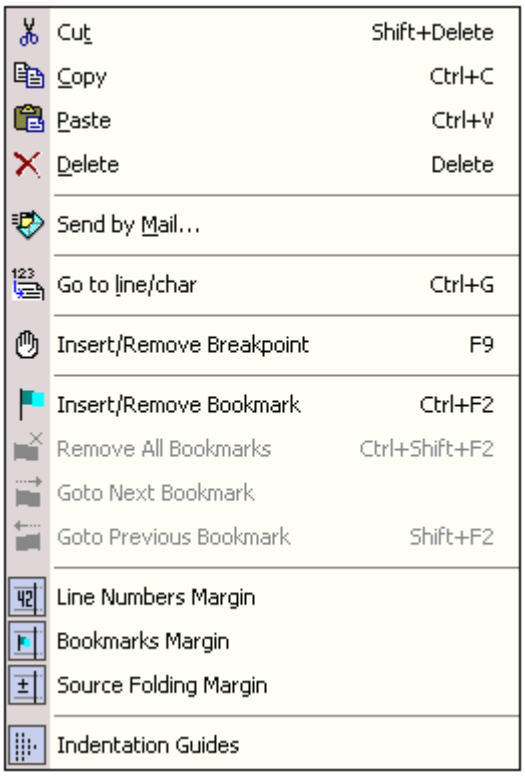
Autokomplettierung
Wenn Sie XML-Dokumente in der Textansicht bearbeiten, wird oft die Wohlgeformtheit eines Dokuments verletzt, so können z.B. unter Umständen End-Tags fehlen, falsch geschrieben sein oder in der Struktur an einer falschen Stelle stehen.
XMLSpy ergänzt den Start- und End-Tag von Elementen automatisch und fügt alle obligatorischen Attribute ein, sobald Sie auf der Tastatur den Elementnamen fertig eingegeben haben. Außerdem wird der Cursor automatisch zwischen den Start- und End-Tag des Elements gesetzt, sodass Sie gleich weitere Child-Elemente oder Inhalt eingeben können:

```
<img src="" alt="">| </img>
```

Mit dem Befehl Wohlgeformtheit prüfen können Sie jederzeit überprüfen, ob das Dokument wohlgeformt ist. Diese Überprüfung wird auch automatisch jedes Mal beim Öffnen oder Speichern eines Dokuments durchgeführt.

Drag-and-Drop und Kontextmenüs
Sie können einen Textblock auch mit Drag-and-Drop an eine andere Stelle verschieben und häufig verwendete Editier-Befehle (wie Ausschneiden, Kopieren, Einfügen, Löschen, Als Mail senden und Gehe zu Zeile/Zeichen) mit einem Rechtsklick direkt über ein Kontextmenü aufrufen.

Die anderen Befehle im Kontextmenü ermöglichen Ihnen Lesezeichen zu setzen/löschen usw. und die Textansicht anzupassen.



Suchen und Ersetzen
Mit Hilfe der Befehle Suchen und Ersetzen können Sie Text schnell finden und ändern. Diese Befehle unterstützen auch Regular Expressions, sodass Ihnen damit leistungsstarke

 © 2007 Altova GmbH

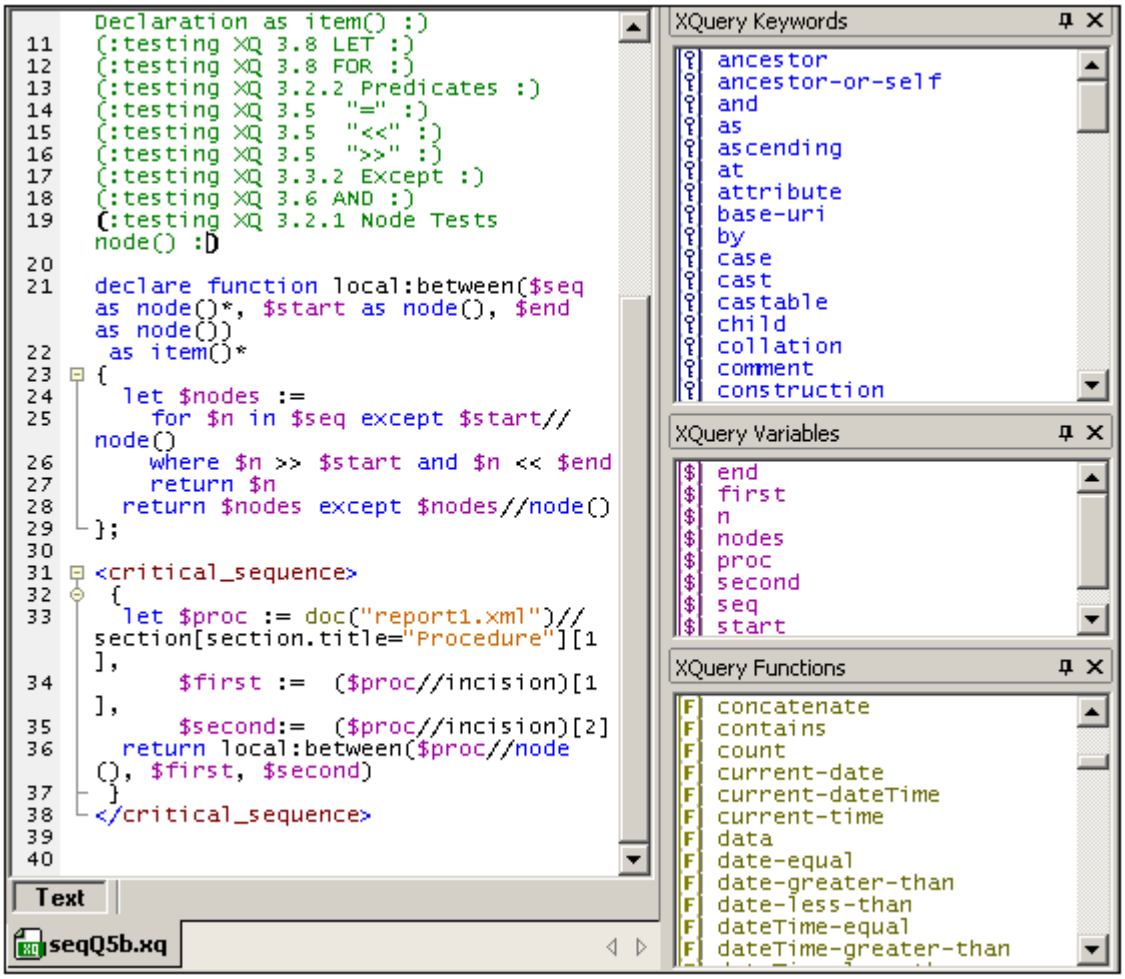
Suchfunktionen zur Verfügung stehen. (Nähere Informationen siehe Bearbeiten | Suchen.)

Unbeschränktes Rückgängigmachen
XMLSpy bietet Ihnen für alle Editiervorgänge unbeschränktes Rückgängigmachen und Wiederherstellen.

Vergößern und Verkleinern
Sie können den Inhalt der Text-Ansicht vergrößern oder verkleinern, indem Sie das Mausrad drehen, während Sie die **Strg**-Taste gedrückt halten. Auf diese Art können Sie den Text in der Text-Ansicht vergrößern oder verkleinern.

3.5 Bearbeiten von XQuery-Dokumenten

In der Textansicht können Sie XQuery-Dokumente bearbeiten. Die Eingabehilfen, Syntaxfärbung und das intelligente Editieren unterscheiden sich dabei von denen, die für die Textansicht von XML-Dokumenten zur Verfügung stehen (*siehe Abbildung unten, Zeilennummerierung und Klappleisten stehen nur in der Enterprise und der Professional Edition zur Verfügung)*. Dieser Textansichtsmodus wird XQuery-Modus genannt. Zusätzlich dazu können Sie Ihr XQuery-Dokument in der Textansicht validieren und den Code in einem XQuery-Dokument (mit einer optionalen XML-Datei, wenn nötig) mit Hilfe des Altova XQuery-Prozessors ausführen .



Hinweis: XQuery-Dateien können nur in der Textansicht bearbeitet werden. Für XQuery-Dateien stehen keine anderen Ansichten zur Verfügung.

Nähere Informationen dazu, wie der Altova-XQuery-Prozessor implementiert ist und XQuery-Dateien verarbeitet, finden Sie unter "Implementierung des XQuery-Prozessors".

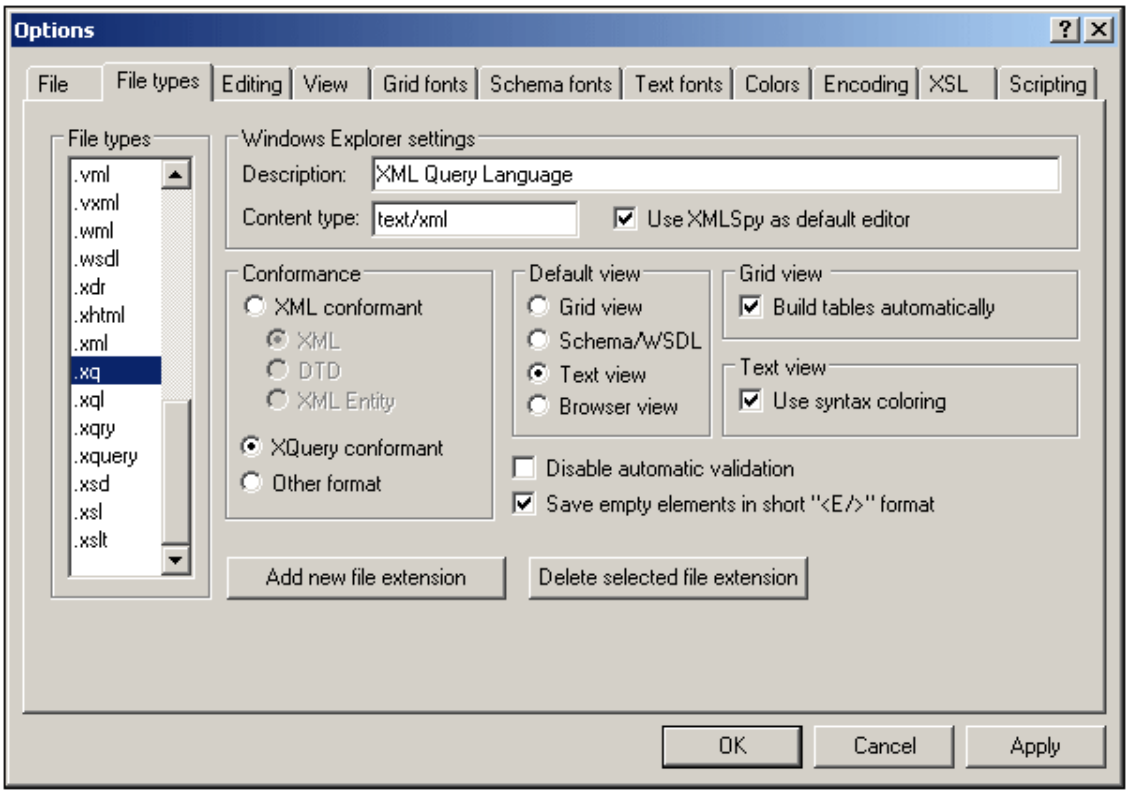
3.5.1 Öffnen eines XQuery-Dokuments

Ein XQuery-Dokument wird automatisch im XQuery-Modus der Textansicht geöffnet, wenn es XQuery-konform ist. Dateien mit der Dateierweiterung . `xq`, . `xql`, und . `xquery` sind in XMLSpy als XQuery-konform vordefiniert.

Definieren weiterer Dateierweiterungen als XQuery-konform

So definieren Sie weitere Dateierweiterungen als XQuery-konform:

1. Wählen Sie **Extras | Optionen**. Daraufhin wird das Dialogfeld "Optionen" angezeigt (*siehe screenshot*).
2. Wählen Sie das Register **Dateiarten**.
3. Klicken Sie auf "Neue Dateierweiterung hinzufügen", um der Liste der Dateiarten eine neue Dateierweiterung hinzuzufügen.
4. Aktivieren Sie unter **Konformität**, den Eintrag **XQuery-konform**.

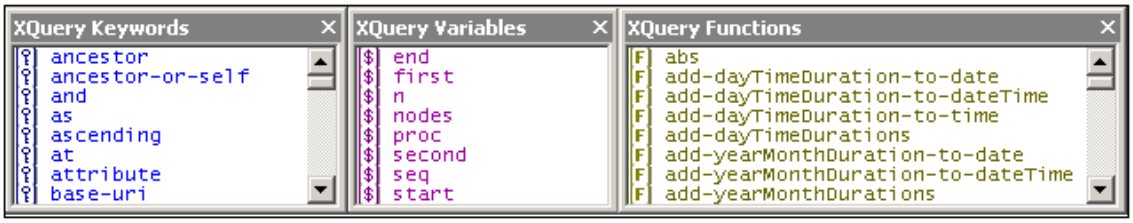


Außerdem sollten Sie in diesem Dialogfeld die folgenden Windows Explorer-Einstellungen vornehmen:

- **Beschreibung**: XML Query Language
- **Inhaltstyp**: Text/xml
- Wenn Sie XMLSpy als Standardeditor für XQuery-Dateien verwenden wollen, aktivieren Sie das Kontrollkästchen **XMLSpy als Standardeditor verwenden**.

3.5.2 XQuery Eingabehilfen

In der Textansicht stehen Ihnen Im XQuery-Modus drei Eingabehilfen zur Verfügung: XQuery-Schlüsselwörter (blau), XQuery-Variablen (violett) und XQuery-Funktionen (olivgrün).



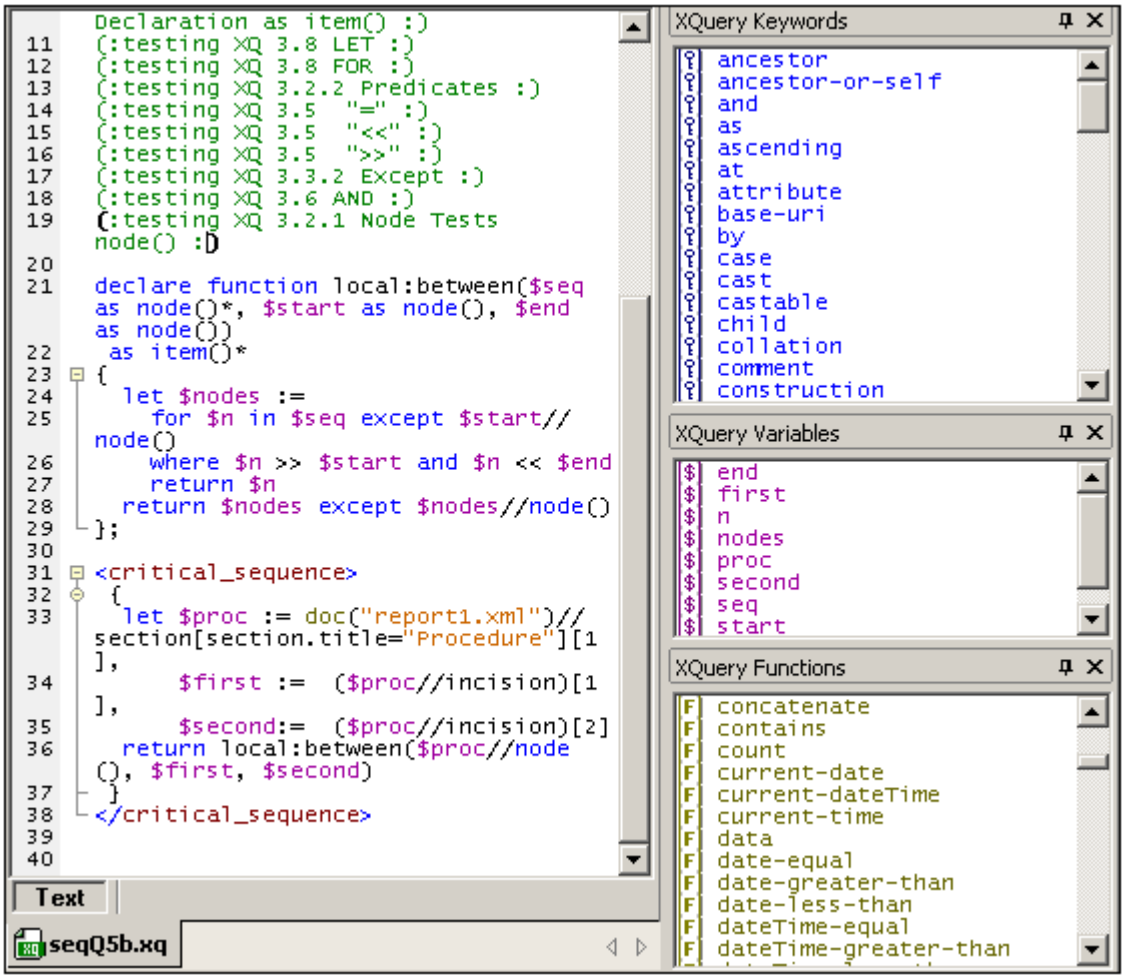
Beachten Sie bitte die folgenden Punkte:

- Die Farbe der Einträge in den drei Eingabehilfen ist unterschiedlich und entspricht der im Text verwendeten Syntaxfärbung. Diese Farben können nicht geändert werden.
- Die aufgelisteten Schlüsselwörter und Funktionen sind diejenigen, die vom Altova XQuery-Prozessor unterstützt werden.
- Die Variablen werden im XQuery-Dokument selbst definiert. Wenn Sie in der Textansicht ein `$`-Zeichen und ein weiteres Zeichen eingeben, erscheint das Zeichen in der Variablen-Eingabehilfe (es sei denn, es existiert eine Variable, die aus genau diesem Zeichen besteht). Sobald ein eingegebener Variablenname mit einem bereits existierenden Variablennamen übereinstimmt, verschwindet der neu eingegebene Variablenname aus der Eingabehilfe.
- Zum Navigieren in einer Eingabehilfe klicken Sie auf einen Eintrag in der Eingabehilfe und bewegen Sie sich mittels Bildlaufleiste, Mausrad oder der Bild auf- bzw. Bild ab-Taste durch die Liste.

Um einen der in den Eingabehilfen aufgelisteten Einträge in das Dokument einzufügen, setzen Sie den Cursor an die gewünschte Stelle und doppelklicken Sie auf den Eintrag. In XQuery stellen einige Zeichenfolgen sowohl ein Schlüsselwort als auch eine Funktion dar (`empty`, `unordered` und `except`). Diese Strings werden immer als Schlüsselwörter eingegeben (blau) - selbst wenn Sie die Funktion dieses Namens in der Funktionen-Eingabehilfe auswählen. Wenn eine Funktion blau angezeigt wird, ist sie durch die Klammern, die auf den Funktionsnamen folgen, als solche zu erkennen.

3.5.3 XQuery Syntaxfärbung

Ein XQuery-Dokument kann sowohl aus XQuery-Code als auch XML-Code bestehen. In diesem Abschnitt wird die Standard-Syntaxfärbung für den XQuery-Code beschrieben. Die Syntaxfärbung für XML-Code in einem XQuery-Dokument ist dieselbe wie die in normalen XML-Dokumenten verwendete. Alle Syntaxfärbungseinstellungen (sowohl für XQuery-Code als auch für XML-Code) werden im Dialogfeld "Optionen" (**Extras | Optionen**) auf dem Register "Textschriftarten" vorgenommen. Beachten Sie, dass XML-Elemente XQuery-Code enthalten können. Der XQuery-Code wird dabei in geschwungene Klammern **{ }** gesetzt (*Beispiel siehe Abbildung; Zeilennummerierung und Klappleiste nur in der Enterprise und der Professional Edition*).



Im XQuery-Modus der Textansicht wird für XQuery-Code die folgende Standard-Syntaxfärbung verwendet:

- `(: Comments einschließlich 'Smiley'-Trennzeichen sind grün :)`
- XQuery-Schlüsselwörter sind blau: `Schlüsselwort`
- XQuery-Variablen, einschließlich des Dollar-Zeichens sind violett: `$start`
- XQuery-Funktionen, **nicht** aber die einschließenden Klammern sind olivgrün: `Funktion()`
- Strings sind orange: `"Prozedur"`
- Der gesamte andere Text, wie z.B. Path-Ausdrücke ist schwarz (*wie der unterstrichene Text unterhalb*). Dies sieht so aus:

```
for $s in doc("report1.xml")//section[section.title =
"Procedure"]
return ($s//incision)[2]/instrument
```

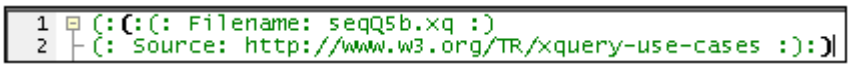
Sie können diese Standardfarben sowie andere Schriftarteinstellungen im Dialogfeld "Optionen" (**Extras | Optionen**) auf dem Register "Textschriftarten" ändern.

Hinweis: Im oben gezeigten Screenshot wird ein farbiges Klammerpaar für einen Comment schwarz und fett angezeigt. Dies geschieht aufgrund der Funktion zum Anzeigen von Kammerpaaren (siehe Intelligentes Editieren von XQuery).

3.5.4 Intelligentes Editieren von XQuery

Im XQuery-Modus der Textansicht stehen Ihnen die folgenden intelligenten Editierfeatures zur Verfügung.

Anzeigen von Klammerpaaren
Die Funktion zum Anzeigen von Klammerpaaren markiert die öffnende und schließende Klammer eines Klammerpaars, sodass der Inhalt eines Klammerpaars klar ersichtlich ist. Dies ist v.a. bei verschachtelten Klammern, wie dies bei XQuery-Comments (*siehe Abbildung unten*) der Fall ist, hilfreich.

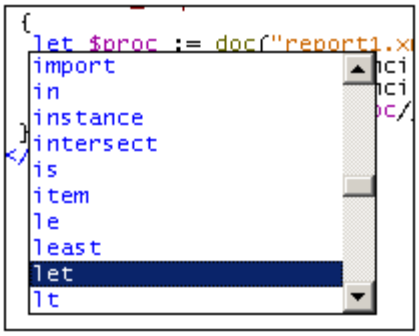


Die Klammerpaaranzeige wird aktiviert, indem Sie den Cursor entweder unmittelbar vor oder unmittelbar hinter eine (öffnende oder schließende) Klammer setzen. Diese Klammer wird zusammen mit ihrem Gegenstück (schwarz und fett) hervorgehoben. Beachten Sie die Position des Cursors im obigen Screenshot.

Die Klammerpaaranzeige ist für runde `( )` , eckige `[ ]` und geschwungene Klammern `{ }` aktiv. Eine Ausnahme bilden spitze Klammern `<>`, die für XML-Tags verwendet werden.

Hinweis: Wenn Sie den Cursor innerhalb eines Start- oder End-Tags eines XML-Elements setzen, wird der andere Teil des Klammerpaars markiert, wenn Sie **Strg+E** drücken. Bei wiederholtem Drücken von **Strg+E** können Sie zwischen dem Start- und End-Tag hin- und herwechseln. Dies ist ein weiteres Hilfsmittel, um Start- und End-Tag eines XML-Elements leichter zu finden.

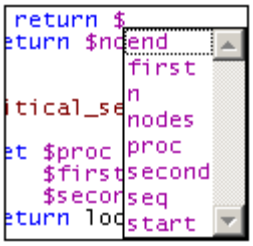
Schlüsselwörter
XQuery-Schlüsselwörter sind in Query-Ausdrücken verwendete Anweisungen und werden blau angezeigt. Wenn Sie den Cursor in oder unmittelbar vor oder hinter ein Schlüsselwort platzieren, wird das Schlüsselwort ausgewählt. Wenn Sie nun **Strg+Leertaste** drücken, wird in einem Popup-Menü eine komplette Liste aller Schlüsselwörter angezeigt. Sie können durch die Liste scrollen und auf ein Schlüsselwort doppelklicken, um das ausgewählte Schlüsselwort zu ersetzen.



In der oben gezeigten Abbildung wurde der Cursor in das Schlüsselwort `let` gesetzt. Wenn Sie auf ein Schlüsselwort aus der Liste doppelklicken, wird das Schlüsselwort `let` durch dieses ersetzt.

Variablen
Variablennamen ist das `$`-Zeichen vorangestellt. Variablen werden violett angezeigt. Die Vorgangsweise bei dieser intelligenten Editierfunktion ist ähnlich wie bei der für Schlüsselwörter. Es gibt zwei Arten, um die Popup-Liste aller Variablen in einem Dokument aufzurufen:

- Geben Sie zuerst ein **$**-Zeichen ein und drücken Sie **Strg+Leertaste**
- Wählen Sie eine Variable aus und drücken Sie **Strg+Leertaste**. (Zum Auswählen einer Variable setzen Sie den Cursor unmittelbar hinter das **$** -Zeichen, in den Namen einer Variable oder unmittelbar hinter den Namen der Variable.)

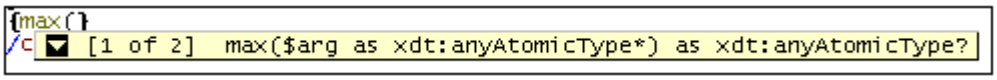


Um eine Variable (während der Eingabe) hinter dem **$**-Zeichen einzufügen oder eine ausgewählte Variable zu ersetzen, doppelklicken Sie im Popup-Menü auf die gewünschte Variable.

Funktionen
Genau wie bei Schlüsselwörtern und Variablen wird hier ein Popup-Menü mit vordefinierten Funktionen angezeigt, wenn Sie eine Funktion (angezeigt in Grün) auswählen und **Strg+Leerzeichen** drücken. (Zum Auswählen einer Funktion setzen Sie den Cursor in einen Funktionsnamen oder unmittelbar vor oder hinter den Namen. Der Cursor darf nicht zwischen die Klammern gesetzt werden, die auf den Funktionsnamen folgen). Wenn Sie im Popup-Menü auf einen Funktionsnamen doppelklicken, wird der ausgewählte Funktionsname durch die Funktion aus dem Popup-Menü ersetzt.

Um einen Tipp anzuzeigen, der die Bezeichnung einer Funktion enthält (*siehe Abbildung unten)* , setzen Sie den Cursor unmittelbar hinter die öffnende Klammer und drücken Sie **Strg+Leertaste.**

Bitte beachten Sie: Die Bezeichnung kann nur für XQuery-Standardfunktionen angezeigt werden.



Der nach unten weisende Pfeil bedeutet, dass es mehrere Funktionen mit demselben Namen gibt, dass diese jedoch unterschiedliche Argumente oder Rückgagetypen haben. Klicken Sie auf die Bezeichnung um die Bezeichnung der nächsten Funktion anzuzeigen (falls vorhanden); wenn Sie mehrmals hintereinander darauf klicken, werden der Reihe nach alle Funktionen dieses Namens angezeigt. Alternativ dazu können Sie auch durch Drücken von **Strg+Umschalt+Nach unten** oder **Strg+Umschalt+nach oben** durch die Funktionen wechseln.

Visuelle Orientierungshilfen
Die Textein- und -ausklappfunktion ist bei geschwungenen XQuery-Klammern, bei XQuery Comments, XML-Elementen und XML-Comments aktiv.

3.5.5 Validierung und Ausführung von XQuery-Dokumenten

Validierung von XQuery-Dokumenten
Um ein XQuery-Dokument zu validieren, gehen Sie vor wie folgt:

1. Machen Sie das XQuery-Dokument zum aktiven Dokument.
2. Wählen Sie **XML | Validieren** oder drücken Sie **F8** oder klicken Sie in der Symbolleiste

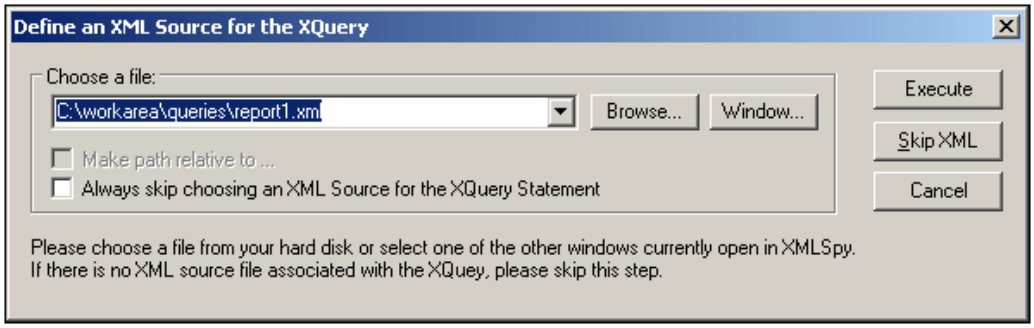
auf das Symbol .

Das Dokument wird daraufhin auf die korrekte XQuery-Syntax überprüft.

Ausführen von XQuery-Dokumenten
XQuery-Dokumente werden in XMLSpy mit dem integrierten XQuery 1.0-Prozessor ausgeführt. Das Resultat wird in einem Fenster in XMLSpy angezeigt.

Normalerweise ist ein XQuery-Dokument nicht mit einem einzelnen XML-Dokument verknüpft, da XQuery-Ausdrücke mit Hilfe der `doc()` -Funktion eine beliebige Anzahl von XML-Dokumenten auswählen können. In XMLSpy können Sie jedoch vor dem Ausführen eines einzelnen XQuery-Dokuments ein XML-Quelldokument dafür auswählen. In diesem Fall ist der Dokument-Node des ausgewählten XML-Quelldokuments der Context-Node, der auf der Root-Ebene des XQuery-Dokument als Ausgangspunkt angegegeben ist. Pfade, die mit einem Schrägstrich beginnen, werden mit diesem Dokument-Node als Context-Element aufgelöst. Um ein XQuery-Dokument auszuführen, gehen Sie vor wie folgt:

1. Machen Sie das XQuery-Dokument zum aktiven Dokument.
2. Wählen Sie **XSL/XQuery | XQuery Transformation** oder klicken Sie in der Symbolleiste auf das Symbol . Daraufhin erscheint das Dialogfeld "Definieren Sie eine XML-Quelle für die XQuery".



3. Wählen Sie eine der folgenden Methoden:
   - Zum Auswählen einer XML-Datei verwenden Sie entweder die Schaltfläche **Durchsuchen** oder die Schaltfläche **Fenster** (damit werden Dateien, die in XMLSpy geöffnet sind und jene die sich in XMLSpy-Projekten befinden, aufgelistet). Wählen Sie eine XML-Quelldatei aus, wenn Sie deren Dokument-Node als Kontextelement für die Root-Ebene des XQuery-Dokuments zuweisen möchten. Klicken Sie auf **Ausführen**.
   - Um dieses Dialogfeld zu überspringen, klicken Sie auf **XML überspringen.**

Das Ergebnisdokument wird als temporäre Datei generiert, die im gewünschten Dateiformat und mit der gewünschten Erweiterung in einem beliebigen Ordner gespeichert werden kann.

Nähere Informationen darüber, wie der Altova XQuery-Prozessor implementiert ist und XQuery-Dateien verarbeitet, finden Sie unter Implementierung des XQuery-Prozessors.

3.5.6 XQuery und XML-Datenbanken

Mit Hilfe eines XQuery-Dokuments können Sie eine Abfrage an einer XML-Datenbank (XML DB) durchführen. Derzeit wird diese XQuery-Funktionalität nur für IBM DB2-Datenbanken unterstützt. Der Vorgang beim Abfragen einer XML-DB mit Hilfe von XQuery läuft folgendermaßen ab: (i) der XQuery-Prozessor wird so eingestellt, dass XML in einer Datenbank

- und nicht XML in einem XML-Dokument - abgefragt wird; und (ii) die XML-Daten in der DB werden aufgerufen.

Dabei müssen die folgenden Schritte durchgeführt werden:

1. Definieren Sie das XQuery-Dokument für die Abfrage der XML-DB, indem Sie am Beginn des Dokuments das Schlüsselwort `XQUERY` einsetzen.
2. Aktivieren Sie (über das Fenster "Info") für das aktive XQuery-Dokument die DB-Unterstützung und stellen Sie (im Dialogfeld "Verbindungsassistent") eine Verbindung zur DB her.
3. Fügen Sie im XQuery-Dokument DB-spezifische XQuery-Erweiterungen ein, um die DB-Daten aufrufen zu können und für XQuery-Operationen verfügbar zu machen.
4. Führen Sie das XQuery-Dokument in XMLSpy aus.

Einrichten des XQuery-Dokuments für die Abfrage der XML DB
Um das XQuery-Dokument für eine Abfrage einer XML DB zu konfigurieren, öffnen Sie das XQuery-Dokument (oder erstellen Sie ein neues XQuery-Dokument) und geben Sie das Schlüsselwort `XQUERY` (Groß- und Kleinschreibung irrelevant) am Beginn des Dokuments (vor dem Prolog) ein; *siehe Beispeile unten*.

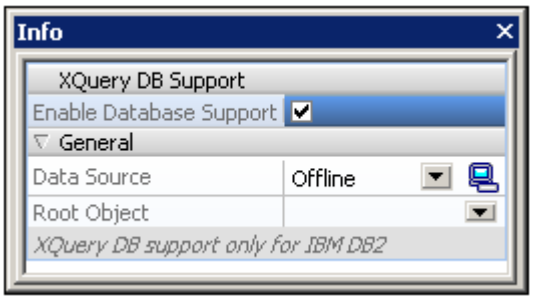
```
XQUERY (: Retrieve details of all customers :)
declare default element namespace "http://www.altova.com/xquery/databases/db2"
;
<a> {db2-fn:xmlcolumn("CUSTOMER.INFO")} </a>
```

Wenn im Dokument der optionale Ausdruck `xquery version` verwendet wird, wird dennoch das Schlüsselwort `XQUERY` benötigt.

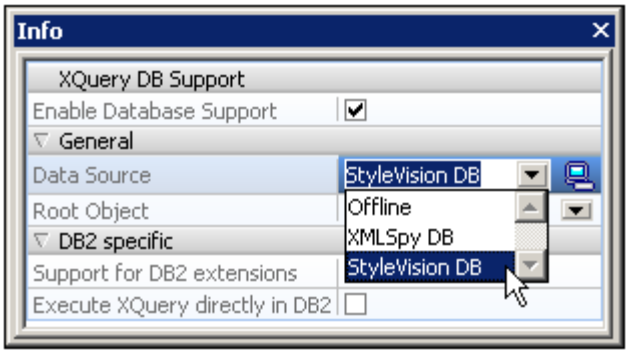
```
XQUERY xquery version "1.0"; (: Retrieve details of all customers :)
declare default element namespace "http://
http://www.altova.com/xquery/databases/db2";
<a> {db2-fn:xmlcolumn("CUSTOMER.INFO")} </a>
```

Anmerkung: Der integrierte XMLSpy XQuery-Prozessor liest das Schlüsselwort `XQUERY` als Indikator dafür, dass eine XML DB aufgerufen werden soll. Folglich führt der Versuch, ein XQuery-Dokument, das das Schlüsselwort `XQUERY` enthält, an einem XML-Dokument auszuführen, das nicht in einer XML DB enthalten ist, zu einem Fehler.

Aktivieren der DB-Unterstützung für XQuery und Herstellen einer Verbindung zur DB
Die DB-Unterstützung für ein XQuery-Dokument wird durch Aktivierung des Kontrollkästchens "Datenbankunterstützung aktivieren" im Fenster "Info" (*Abbildung unten)* aktiviert. Beachten Sie: Die Unterstützung von DB muss für jedes XQuery-Dokument separat aktiviert werden. Jedes Mal muss ein XQuery-Dokument neu geöffnet werden.



Wenn Sie die DB-Unterstützung im Fenster "Info" aktivieren, erscheint ein Verbindungsassistent, mit Hilfe dessen Sie eine Verbindung zu einer Datenbank herstellen können. Derzeit werden nur IBM DB2-Datenbanken unterstützt. Die Herstellung einer Verbindung zu einer DB ist im Abschnitt Herstellen einer Verbindung zu einer Datenquelle beschrieben. Wenn bereits Verbindungen zu Datenquellen bestehen, werden diese im Fenster "Info" in der Auswahlliste "Datenquellen" aufgelistet (*Abbildung unten*) und Sie können eine dieser Datenquellen als Datenquelle für das aktive XQuery-Dokument auswählen. Im Fenster "Info" können Sie auch das Root-Objekt aus den verfügbaren Root-Objekten in der Auswahlliste auswählen.



Der Verbindungsassistent (mit Hilfe dessen Sie eine Verbindung zu einer DB herstellen können), kann jederzeit durch Klicken auf die Schaltfläche im Fenster "Info" aufgerufen werden.

Anmerkung: Beim Schließen eines XQuery-Dokuments geht auch die Verbindung zur Datenbank verloren. Wenn Sie das XQuery-Dokument später wieder öffnen, müssen Sie auch die DB-Verbindung wieder herstellen.

IBM DB2-spezifische Erweiterungen für XQuery
Zum Abrufen von Daten aus einer IBM DB2-Datenbank können in XQuery-Dokumenten zwei IBM DB2-spezifische Funktionen verwendet werden:

- `db2-fn:xmlcolumn` ruft eine gesamte XML-Spalte ab, ohne Sie zu durchsuchen oder zu filtern.
- `db2-fn:sqlquery` ruft Werte auf Basis einer SQL `SELECT`-Anweisung ab

Sie können anschließend mit den XML-Daten, die mit Hilfe dieser Funktionen abgerufenen wurden, mit XQuery-Standardkonstrukten arbeiten. *Siehe Beispiele unten*.

`db2-fn:xmlcolumn`: Das Argument der Funktion ist ein String-Literal, bei dem die Groß- und Kleinschreibung beachtet werden muss, das eine XML-Spalte in einer Tabelle ermittelt. Das String-Literal-Argument muss ein qualifizierter Spaltenname von Typ XML sein. Die Funktion gibt alle XML-Daten in der Spalte als Sequenz zurück, ohne daran eine Suchbedingung anzuwenden. Im folgenden Beispiel werden alle Daten der (XML)-Spalte `INFO` der Tabelle `CUSTOMER` innerhalb eines `<newdocelement>` Elements der obersten Ebene zurückgegeben:

```
XQUERY (: Retrieve details of all customers :)
declare default element namespace "http://www.altova.com/xquery/databases/db2"
;
<newdocelement> {db2-fn:xmlcolumn("CUSTOMER.INFO")} </newdocelement>
```

Die abgerufenen Daten können anschließend mit XQuery-Konstrukten abgefragt werden. Im Beispiel unten werden die XML-Daten, die aus der XML-Spalte `INFO` der Tabelle `CUSTOMER` abgerufen wurden, mit Hilfe eines XQuery-Konstrukts gefiltert, sodass nur die Profile der

Kunden aus Toronto abgerufen werden.

```
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>
```

Anmerkung: Im Beispiel oben ist das Dokument-Element der XML-Dateien in den einzelnen Zellen das Element `customerinfo`, und als Root-Node der XML-Sequenz, die von `db2-fn:xmlcolumn` zurückgegeben wird, wird ein abstrakter Node oberhalb der `customerinfo` Nodes angenommen.

`db2-fn:sqlquery`: Die Funktion verwendet eine SQL Select-Anweisung als Argument und gibt eine Sequenz von XML-Werten zurück. Die abgerufene Sequenz wird anschließend mit XQuery-Konstrukten abgefragt. Im folgenden Beispiel wird die Spalte `INFO` in der Tabelle `CUSTOMER` nach Datensätzen durchsucht, deren `CID`-Feld einen Wert zwischen `1000` und `1004` hat.

```
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>
```

Die oben angeführte XQuery gibt die folgende Ausgabe zurück:

```
<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>
```

Beachten Sie die folgenden Punkte:

- Die Standard-Element-Namespace-Deklaration im Prolog gilt für das gesamte XQuery-Dokument und dient sowohl zur Navigation im XML-Dokument als auch zur Erstellung neuer Elemente. D.h. der XQuery-Selektor `name` wird zu `<default-element-namespace>:name` erweitert und Elemente, die erstellt werden, wie z.B. `persons`, befinden sich im Standard-Element-Namespace.
- Bei der SQL Select-Anweisung wird die Groß- und Kleinschreibung nicht beachtet.
- Die `WHERE`-Klausel der Select-Anweisung sollte ein weiteres Datenbankelement referenzieren - keinen Node innerhalb der aufgerufenen XML-Datei.
- Der Schrägstrich "`/`" hinter der Funktion `db2-fn:sqlquery` steht für das erste Datenelement der zurückgegebenen Sequenz und dieses Datenelement ist der Kontext-Node für die weitere Navigation.

Ausführen des XQuery-Dokuments
Um das XQuery-Dokument auszuführen, wählen Sie den Befehl **XQuery-Ausführung** (Menü **XSL/XQuery**). Alternativ dazu können Sie auch **Alt+F10** drücken oder auf die Schaltfläche "XQuery-Ausführung" klicken. Das Ergebnis der Ausführung wird in einem neuen Dokument angezeigt.

© 2007 Altova GmbH

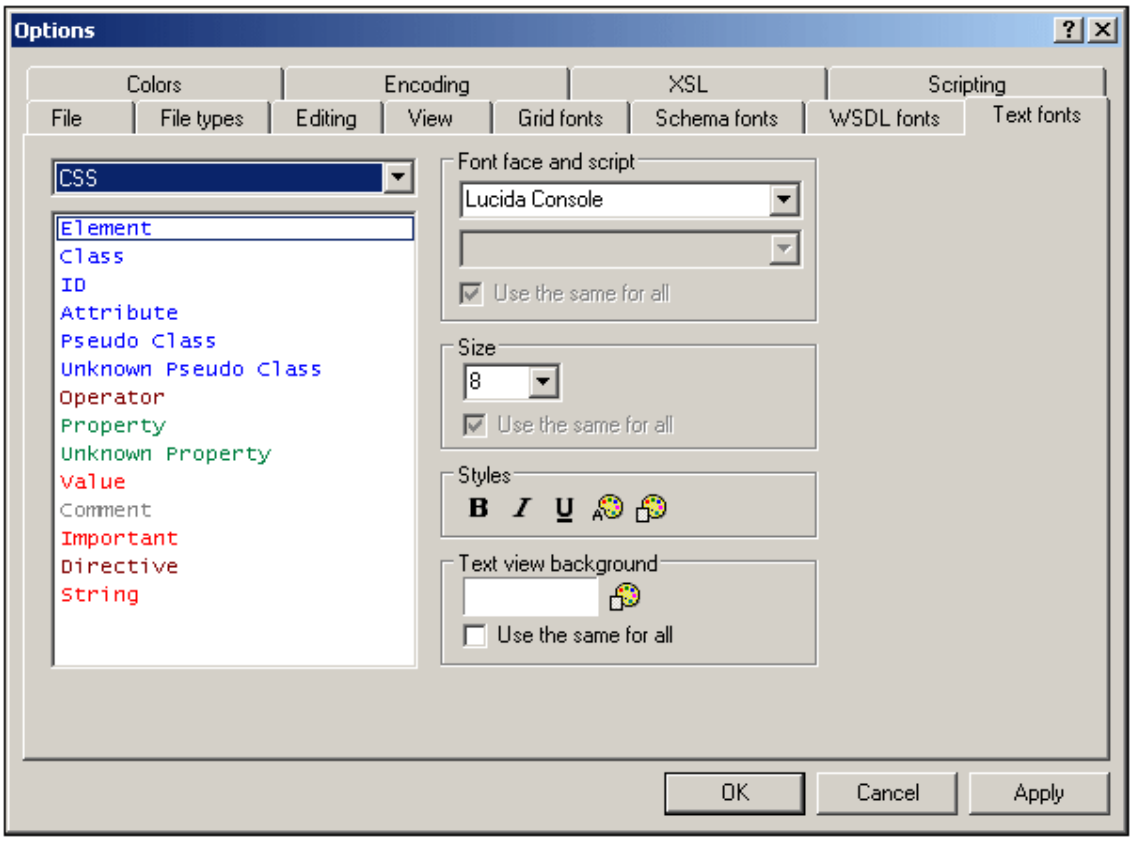
3.6 Bearbeiten von CSS-Dokumenten

CSS-Dokumente können mit Hilfe der intelligenten Bearbeitungsfunktionen der Textansicht bearbeitet werden. Diese Funktionen werden im Zusammenhang mit der Bearbeitung von CSS-Dokumenten im Folgenden beschrieben.

Syntaxfärbung: Eine CSS-Regel besteht aus einem Selektor, einer oder mehreren Eigenschaften und den Werten dieser Eigenschaften. Diese drei Komponenten können in noch spezifischere Kategorien unterteilt werden; so kann es sich bei einem Selektor z.B. um eine Klasse, eine Pseudo-Klasse, eine ID, ein Element oder ein Attribut handeln. Außerdem kann ein CSS-Dokument zusätzlich zu den Regeln noch weitere Elemente enthalten, wie z.B. Kommentare. In der Textansicht kann jede dieser Kategorien in einer anderen Farbe angezeigt werden (*Abbildung unten*), je nachdem, welche Einstellungen Sie im Dialogfeld "Optionen" (*siehe unten*) vorgenommen haben.

```
.header
{
    font-family: "Arial", sans-serif;
    font-weight: bold;
    color: red;
}
```

Sie könnnen die Farben der verschiedenen CSS-Komponenten im Dialogfeld "Optionen" (*Abbildung unten*) auf dem Register "Textschriftarten" definieren. Wählen Sie in der Auswahlliste links oben die Option "CSS" aus und wählen Sie anschließend im Bereich "Stile" für die einzelnen CSS-Elemente die gewünschte Farbe aus.

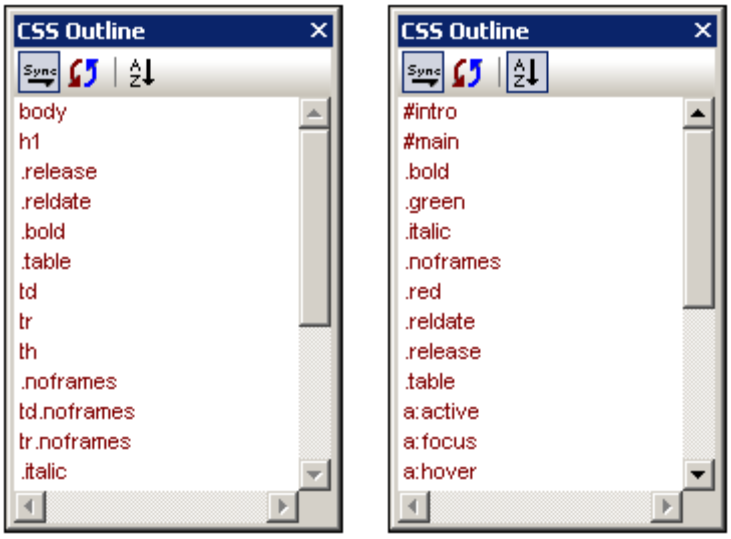


Klappleisten: Jede Regel kann erweitert und reduziert angezeigt werden, indem Sie links von

der Regel (*siehe Abbildung oben*) auf das Plus- bzw. Minus-Symbol klicken. Beachten Sie außerdem, dass die geschwungenen Klammern, innerhalb derer eine Regel angezeigt wird (*Abbildung oben*), fett angezeigt werden, wenn der Cursor entweder vor oder hinter eine der geschwungenen Klammern platziert wird. Auf diese Art sehen Sie genau, wo die Definition einer bestimmten Regel beginnt und wo sie endet.

CSS Outline: Die Eingabehilfe "CSS Outline" (*Abbildungen unten*) bietet einen Überblick über die in Gruppen aufgelisteten Selektoren des Dokuments. Jede Gruppe kann reduziert und erweitert angezeigt werden. Wenn Sie unter CSS Outline auf einen Selektor klicken, wird er im Dokument markiert. In der linken unteren Abbildung sind die Selektoren nicht sortiert und werden in der Reihenfolge aufgelistet, in der sie im Dokument vorkommen. In der Abbildung auf der rechten Seite wurde die Funktion zur alphabetischen Sortierung (über die Symbolleistenschaltfläche) aktiviert, und die Selektoren sind alphabetisch sortiert.

Beachten Sie bitte die folgenden Punkte: (i) Bei der Überprüfung der alphabetischen Reihenfolge der Selektoren werden alle Teile des Selektors berücksichtigt, einschließlich Punkt, Raute und Doppelpunkte; (ii) Wenn mehrere Selektoren zu einer einzigen Regel zusammengruppiert werden (z.B.. `h4, h5, h6 {...}`), wird jeder Selektor in der Gruppe separat aufgelistet.



Die Symbolleisten-Schaltflächen in der Eingabehilfe "CSS Outline" (von links nach rechts), haben die folgenden Funktionen:

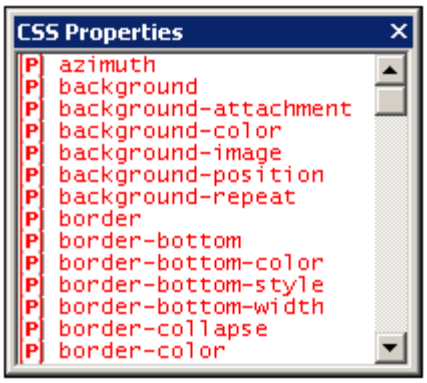
Schaltet die automatische Synchronisierung (mit dem Dokument) ein bzw. aus. Wenn die automatische Synchronisierung aktiviert ist, werden die Selektoren schon während der Eingabe ins Dokument in die Eingabehilfe eingegeben.

Synchronisiert die Eingabehilfe mit dem aktuellen Zustand des Dokuments.

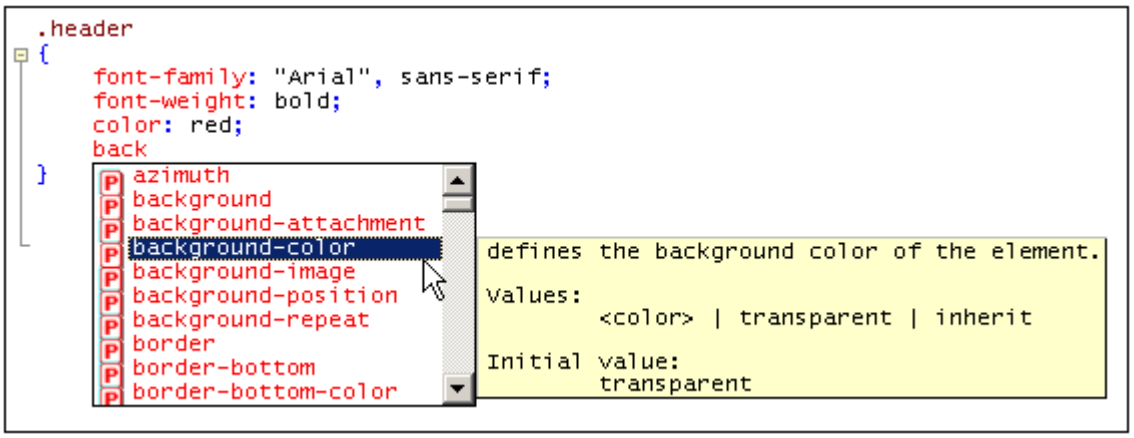
Schaltet die alphabetische Sortierung ein bzw. aus. Wenn diese Sortierung deaktiviert ist, werden die Selektoren in der Reihenfolge aufgelistet, in der sie im Dokument aufscheinen. Bei alphabetischer Sortierung werden ID-Selektoren als erste angezeigt, da ihnen ein Rautensymbol vorangestellt ist (z. B. `#intro`).

© 2007 Altova GmbH

Eingabehilfe "Eigenschaften": Die Eingabehilfe "Eigenschaften" (*Abbildung unten*) bietet eine Liste aller CSS-Eigenschaften in alphabetischer Reihenfolge. Eine Eigenschaft kann durch Doppelklick auf die Eigenschaft an der Cursorposition eingefügt werden.



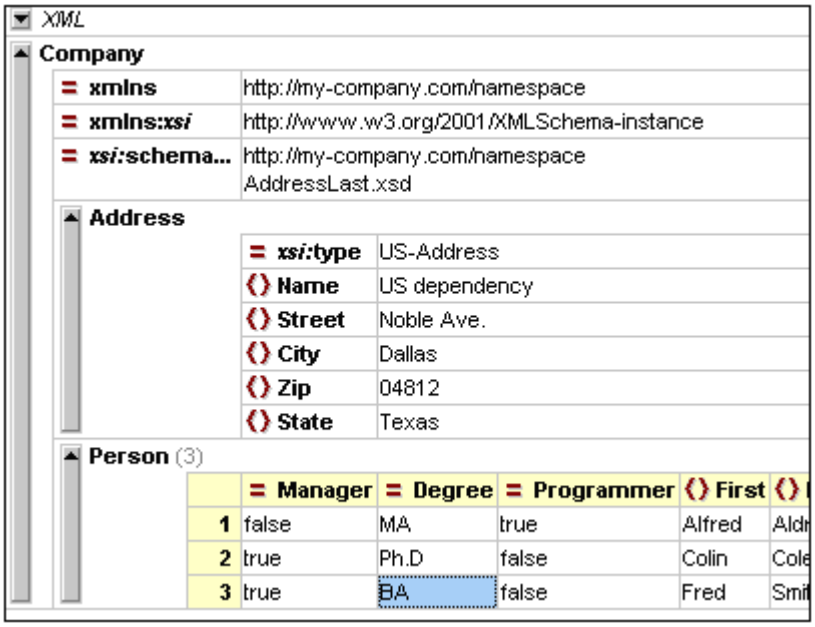
Autokomplettierung von Eigenschaften und Tooltipps für Eigenschaften: Während Sie den Namen einer Eigenschaft eingeben, wird eine Liste von Eigenschaften eingeblendet, die mit diesen Buchstaben beginnen (*Abbildung unten*). Alternativ dazu können Sie den Cursor auch an eine beliebige Stelle innerhalb eines Eigenschaftsnamens platzieren und anschließend **Strg+Leertaste** drücken, um die Liste der CSS-Eigenschaften aufzurufen.



Sie können einen Tooltipp mit der Definition der Eigenschaft und ihren möglichen Werten anzeigen, indem Sie die Liste hinunterscrollen oder mit der Nach oben- und Nach unten-Taste Ihrer Tastatur zu der Eigenschaft navigieren. Der Tooltipp der markierten Eigenschaft wird angezeigt. Um eine Eigenschaft einzufügen, drücken Sie entweder die Eingabetaste, wenn die Eigenschaft ausgewählt ist, oder klicken Sie darauf.

4 Enhanced Grid-Ansicht

Durch Klicken auf die Schaltfläche **Grid** am unteren Rand des Dokumentenfensters wird die Enhanced Grid-Ansicht (oder Grid-Ansicht) geöffnet. Sie zeigt die hierarchische Struktur von XML-Dokumenten mittels verschachtelter Datenbehälter an. Diese können für eine bessere Übersicht über die gesamte Dokumentstruktur erweitert oder reduziert werden. In der Grid-Ansicht können Sie Inhalt und Struktur eines Dokuments einfach bearbeiten.



Ein hierarchisch dargestelltes Objekt (z.B. XML-Deklaration, Dokumenttyp-Deklaration oder ein Element, das Child-Elemente enthält), wird durch eine graue senkrechte Leiste auf der linken Seite dargestellt, die einen kleinen nach oben weisenden Pfeil enthält. Durch Klicken auf die Leiste können Sie das Datenelement erweitern oder reduzieren. Ein Element wird durch das Symbol gekennzeichnet, ein Attribut durch das Symbol .

Anzeige und Navigation

- Der Inhalt eines hierarchischen Elements hängt von dessen Eigenschaften ab. Bei Elementen handelt es sich dabei z.B. normalerweise um Attribute, Character Data, Comments und Child-Elemente. Attribute werden immer aufgelistet und erscheinen in der Reihenfolge der Input-Datei. Den Attributen folgen alle weiteren Objekte genau in der Reihenfolge, in der diese in der Quelldatei angegeben wurden. Sie können beliebig mittels Drag & Drop neu arrangiert werden. Diese Änderung wird auch in den Quelldaten übernommen.
- Falls ein Element nur Character Data enthält, werden diese Daten in der Zeile angezeigt, in der sich auch das Element befindet. Dieses Element wird dann nicht als hierarchisch angesehen. Character Data für andere Elemente werden eingerückt (zusammen mit Attributen und eventuellen Child-Elementen) und als "Text" gekennzeichnet.
- Bei reduzierten Elementen werden ihre Attribute und deren Werte in derselben Zeile in Grau angezeigt. Diese Attribut-Vorschau ist besonders hilfreich beim Editieren von XML-Dokumenten mit einer großen Anzahl von Elementen desselben Namens, die sich nur durch Inhalt oder Attribute unterscheiden (z.B. Datenbank-Applikationen).
- Mit Hilfe der Pfeiltasten können Sie die Laufleiste in der Grid-Ansicht bewegen.

© 2007 Altova GmbH

- Mit Hilfe der + und – Tasten der Zahlentastatur können Sie Datenelemente erweitern und reduzieren.

Anpassen der Grid-Ansicht

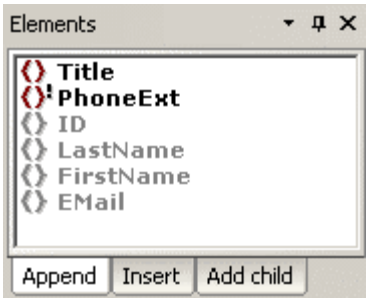
- Um die Größe von Spalten anzupassen, platzieren Sie den Cursor über den entsprechenden Rahmen und ziehen Sie diesen an die gewünschte Stelle.
- Um die Größe einer Spalte an die Breite des breitesten Eintrags anzupassen, doppelklicken Sie auf den rechten Rand der gewünschten Spalte.
- Um die Spaltenbreite so anzupassen, dass der gesamte Inhalt zu sehen ist, wählen Sie den Menüeintrag Ansicht | Optimale Breite oder klicken Sie auf das Symbol "Optimale Breite" .
- Die Höhe der Zellen ist vom jeweiligen Inhalt abhängig. Sie kann über die Menüoption **Extras | Optionen | Ansicht | Enhanced Grid-Ansicht** "Max. Zellenhöhe xxx Zeilen" angepasst werden.

Bitte beachten Sie: Wenn Sie Daten in der Grid-Ansicht markieren und in die Textansicht wechseln, werden diese auch in der Textansicht markiert angezeigt.

Intelligentes Editieren
Die Grid-Ansicht gestattet intelligentes Editieren, wenn das XML-Dokument auf einem Schema basiert. Nähere Informationen dazu finden Sie unter Editieren in der Grid-Ansicht. Die in der Grid-Ansicht verfügbaren Eingabehilfen sind im Folgenden beschrieben.

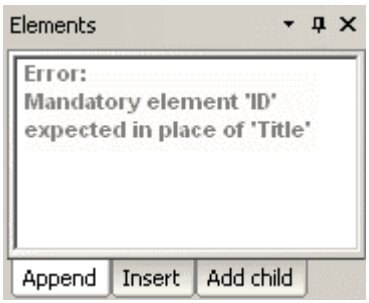
Eingebettete Datenbank/Tabellen-Ansicht
In der Grid-Ansicht können sich wiederholende Elemente in einer Datenbank/Tabellen-Ansicht angezeigt werden. Diese Funktion kann zum Editieren von jeder beliebigen XML, XSD, XSLT usw. Datei verwendet werden. Die in der Datenbank/Tabellen-Ansicht verwendeten Eingabehilfen sind weiter unten beschrieben.

Element-Eingabehilfe
In der Grid-Ansicht hat die Element-Eingabehilfe drei Register: Anhängen, Einfügen und Child hinzufügen. Auf dem Register **Anhängen** werden Elemente angezeigt, die nach allen Geschwisterelementen des aktuellen Elements angehängt werden können; auf dem Register **Einfügen** werden alle Elemente angezeigt, die vor dem aktuellen Element eingefügt werden können und auf dem Register **Child hinzufügen** werden alle jene Elemente angezeigt, die als Child-Element des aktuellen Elements eingefügt werden können.



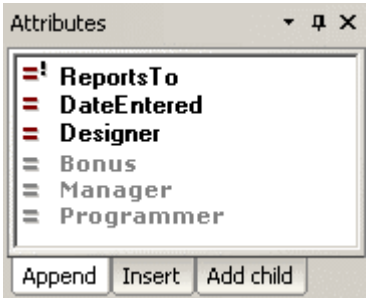
Um ein Element einzufügen, wählen Sie das entsprechende Register und doppelklicken Sie auf das gewünschte Element. Obligatorische Elemente werden durch ein vorangestelltes Ausrufezeichen gekennzeichnet. Geschwisterelemente von zulässigen Elementen, die selbst an der Cursorposition nicht eingefügt werden können, sind ausgegraut und deaktiviert.

Wenn Sie eine Struktur erstellen, die nicht dem in Ihrem Schema definierten Content Model entspricht, zeigt der integrierte Validierungsparser im Element-Eingabehilfefenster der Grid-Ansicht eine Fehlermeldung an.



Bitte beachten Sie: Im Dialogfeld **Optionen** (**Extras | Optionen | Bearbeiten**) können Sie festlegen, dass obligatorische Child-Elemente eingefügt werden können, wenn ein Element eingefügt wird.

Attribut-Eingabehilfe
Die Attribut-Eingabehilfe enthält eine Liste aller für das derzeit bearbeitete Element verfügbaren Attribute. Obligatorische Attribute sind durch ein Ausrufezeichen "**!**" vor dem Namen des Attributs gekennzeichnet. Wurde für dieses Element bereits ein Attribut angegeben, wird dieses Attribut grau angezeigt.



- Um die Attribute auf den Registern "Anhängen" und "Einfügen" zu verwenden, wählen Sie in der Grid-Ansicht ein existierendes Attribut oder ein Element, das ein Child-Element des Parent-Elements des Attributs ist, aus und doppelklicken Sie in der Eingabehilfe auf das gewünschte Attribut.
- Um die Attribute aus dem Register "Child hinzufügen" zu verwenden, wählen Sie in der Grid-Ansicht das Parent-Element des Attributs und doppelklicken Sie in der Eingabehilfe auf das gewünschte Attribut.

Bitte beachten Sie: Existierende Attribute, die bereits hinzugefügt wurden und zu diesem Element kein zweites Mal hinzugefügt werden dürfen, werden grau angezeigt.

Entities-Eingabehilfe
In der Entities-Eingabehilfe werden alle geparsten oder nicht geparsten Entities angezeigt, die innerhalb der Zeile oder in einer externen DTD deklariert sind. Wenn in der Grid-Ansicht ein Text- oder Attribut-Node ausgewählt ist, ist das Register "Child hinzufügen" leer, da solche Nodes laut Definition keine Child-Elemente enthalten können.

Um in der Grid-Ansicht eine Entity einzufügen, platzieren Sie den Cursor an der gewünschten Position im Textfeld oder wählen Sie das gewünschte Feld aus; wählen Sie anschließend das entsprechende Register aus und doppelklicken Sie auf die Entity. Beachten Sie die folgenden Regeln:

- Wenn der Cursor in ein Textfeld (z.B. auch ein Attributwertfeld) gesetzt wird, wird die Entity an der Cursorposition eingefügt.
- Ist ein Element mit einem nur-Text-Child-Element (d.h. `#PCDATA` oder `simpleType`) ausgewählt und der Cursor befindet sich nicht im Textfeld, wird vorhandener Text durch

 © 2007 Altova GmbH

die Entity **ersetzt.**

- Wenn ein nicht-Text-Feld ausgewählt ist, wird die Entity an einer Stelle als Text erstellt, die dem ausgewählten Register der Eingabehilfe entspricht.

Bitte beachten Sie: Wenn Sie eine interne Entity hinzufügen, müssen Sie Ihr Dokument speichern und erneut öffnen, damit die Entity in der Entities-Eingabehilfe angezeigt wird.

4.1 Editieren in der Grid-Ansicht

Beim Editieren eines schemabasierten (DTD oder XML-Schema) XML-Dokuments in der Enhanced Grid-Ansicht stehen Ihnen auf Basis von aus dem Schema stammenden Informationen intelligente Editierfeatures zur Verfügung. Diese Features sind im Folgenden aufgelistet.

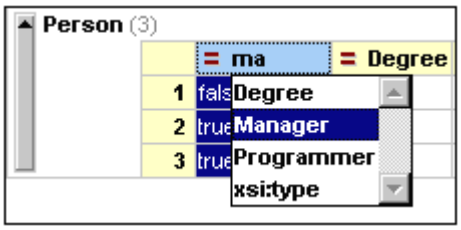
Einfügen oder Anhängen von Element- oder Attributnamen
Zum Einfügen oder Anhängen eines Attributs oder Elements relativ zum ausgewählten Datenelement, können Sie entweder die Befehle im XML-Menü oder die Symbole in der **Attribut- und Element-Symbolleiste** verwenden. Falls diese Symbolleiste nicht sichtbar ist, wählen Sie die Menüoption **Extras | Anpassen** und aktivieren Sie auf dem Register **Symbolleisten** den Eintrag **Attr & Elem**. Eine ausführliche Beschreibung zur Verwendung der Befehle der Symbolleiste und des XML-Menüs finden Sie im Abschnitt Menü "XML".

Um ein Element oder Attribut einzufügen oder anzuhängen, gehen Sie folgendermaßen vor:

1. Wählen Sie das Datenelement, relativ zu welchem das Element oder Attribut eingefügt oder angehängt werden soll.
2. Klicken Sie auf das entsprechende Symbol in der Element- und Attributsymbolleiste oder wählen Sie den Befehl im XML-Menü aus. Daraufhin wird im Raster ein neuer Eintrag erstellt und es erscheint ein Popup-Fenster mit verfügbaren Element- oder Attributoptionen.
3 Wählen Sie im Popup-Fenster das gewünschte Element oder Attribut aus und bestätigen Sie die Auswahl durch Klicken oder Drücken der Eingabetaste. Alternativ dazu können Sie den Namen des gewünschten Elements oder Attributs auch eingeben.

Sie werden feststellen, dass die verschiedenen Eingabehilfen je nachdem, was in der Grid-Ansicht gerade ausgewählt ist, ständig aktualisiert werden. Im Info-Fenster werden jeweils wichtige Informationen zum ausgewählten Element/Attribut angezeigt.

Bearbeiten eines Element- oder Attributnamens
Wenn Sie den Namen eines Elements oder Attributs bearbeiten, wird ein Popup-Menü mit den verfügbaren Optionen geöffnet. Diese Optionen sind abhängig von der Position des Elements und dem im Schema definierten Content Model. Ein ähnliches Popup-Menü wird angezeigt, wenn der Inhalt eines Elements oder Attributs durch eine Enumeration oder Choice eingeschränkt wird.



So bearbeiten Sie einen Element- oder Attributnamen:

1. Doppelklicken Sie auf den Element- oder Attributnamen. Es erscheint ein Popup-Fenster mit den verfügbaren Optionen.
2. Wählen Sie das gewünschte Element oder Attribut aus dem Popup-Menü aus.
3. Bestätigen Sie die Auswahl durch Drücken der **Eingabetaste** oder Klicken auf den Namen. (mit **Esc** verwerfen Sie die Änderungen.)

© 2007 Altova GmbH

Kontextmenü der Grid-Ansicht
Zusätzlich zu den Befehlen, die über die verschiedenen Menüs und Symbolleisten zur Verfügung stehen, bietet die Grid-Ansicht auch ein Kontextmenü, in dem die am häufigsten verwendeten Befehle aus den verschiedensten Menüs zusammengefasst sind. Klicken Sie mit der rechten Maustaste auf das Datenelement, für das Sie eine Aktion durchführen wollen, um das Kontextmenü zu öffnen.

4.2 Datenbank/Tabellen-Ansicht

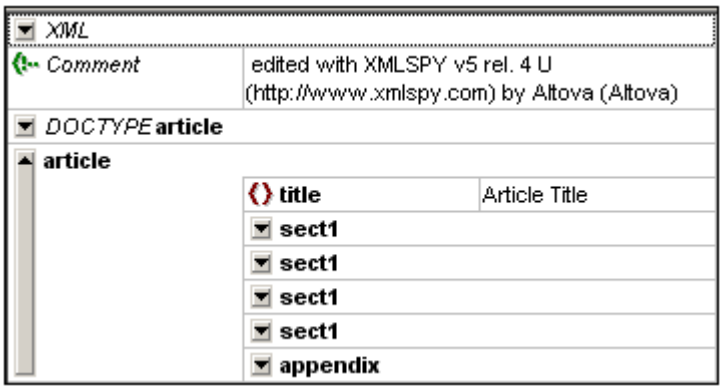
Die **Datenbank/Tabellen-Ansicht** (im Folgenden Tabellenansicht genannt) ist in die Enhanced Grid-Ansicht integriert. Sich wiederholende Elemente werden in dieser Ansicht in einem Tabellenformat angezeigt. Die Tabellenansicht unterscheidet sich von der normalen Grid-Ansicht insofern, als darin für jeden Child-Typ des in Tabellenform dargestellten Elements eine Spalte angelegt wird. Sie können anschließend für ganze Spalten oder Auswahlbereiche Eigenschaften ändern.

Sehen Sie sich als Beispiel das folgende XML-Dokument an:

```
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>
```

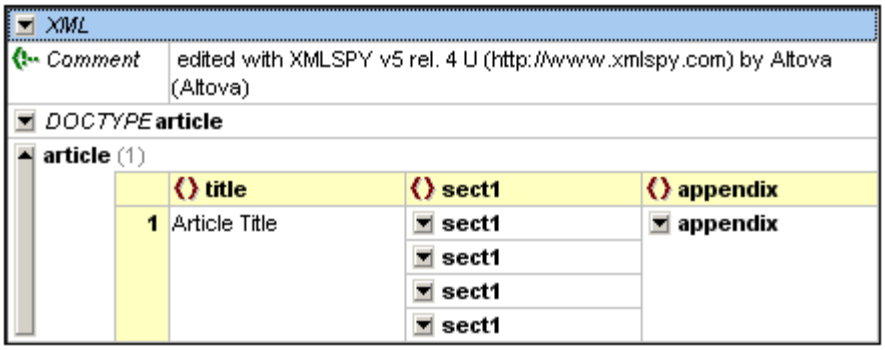
Das Dokument-Element ist `article` und `article` hat die folgende Sequenz von Child-Elementen: ein `title`-Element, vier `sect1`-Elemente und ein `appendix`-Element. Jedes `sect1`- und `appendix`-Element hat ein `title`-Element gefolgt von einer Reihe von `para` oder `simpara`-Elementen.

In der normalen Grid-Ansicht wird dieses Dokument folgendermaßen dargestellt:

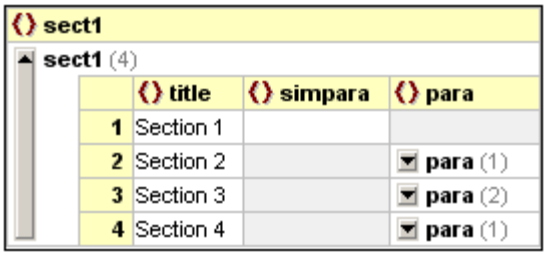


Hier sehen Sie nun die Tabellenansicht dieses Dokuments - besser gesagt des Elements `article`. (Um zu dieser Ansicht zu gelangen, wählen Sie das Element `article` in der

normalen Grid-Ansicht aus (durch Anklicken) und klicken Sie auf das Symbol **Als Tabelle anzeigen** . Alternativ dazu wählen Sie den Menübefehl **XML | Tabelle | Als Tabelle anzeigen (F12)**.)



Beachten Sie, dass jedem Child-Element von `article` (dem als Tabelle angezeigten Element) - also `title`, `sect1` und `appendix` - eine Spalte zugewiesen wurde und dass jede Instanz jedes Child-Typs in der entsprechenden Spalte aufgelistet ist. Beachten Sie außerdem, dass die Tabellenstruktur nur bis zur Ebene der Child-Elemente geht (die `sect1`-Elemente selbst werden nicht als Tabelle dargestellt). Um die `sect1`-Elemente als Tabelle darzustellen, wählen Sie eines der `sect1`-Elemente in der Spalte `sect1` aus und klicken Sie auf . Die `sect1`-Elemente werden nun in Tabellenform dargestellt (siehe Abbildung unten): Jedem der Child-Elemente wird eine Spalte in der `sect1`-Tabelle zugewiesen.



Wenn es für eine der vier Instanzen von `sect1` ein Child-Element gibt (`title`, `simpara` oder `para`), so hat die entsprechende Zelle einen weißen Hintergrund (z.B. `simpara` in der ersten Instanz von `sect1`). Wenn es für eine Instanz kein Child-Element gibt, so hat die entsprechende Zelle einen grauen Hintergrund (z.B. `para` in der ersten Instanz von `sect1`). Daraus ersieht man, dass für jeden unterschiedlichen Child-Typ aller `sect1`-Instanzen eine eigene Spalte angelegt wurde.

Bitte beachten Sie:

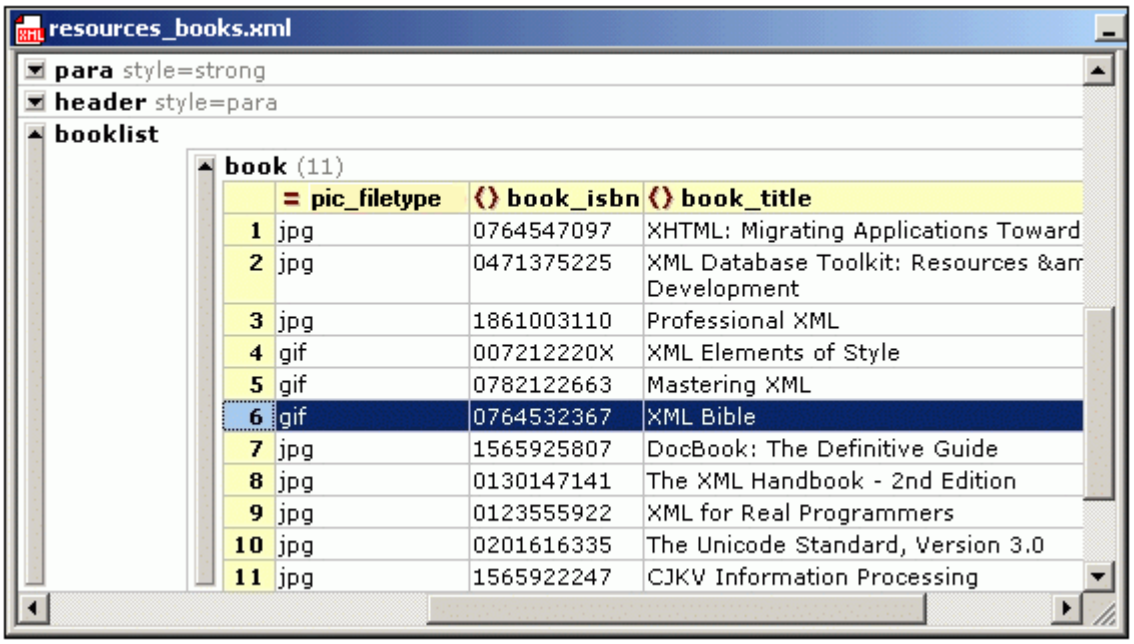
- Auch Attribute werden als Child Nodes betrachtet. Auch für Attribute werden Spalten erstellt.
- Sie können zwischen der normalen Grid-Ansicht und der Tabellenansicht wechseln, indem Sie das gewünschte Element auswählen und auf klicken oder **F12** drücken.
- Wenn Sie das Dokument in der Tabellenansicht ansehen und in die Textansicht wechseln, wird es beim Zurückwechseln in die Grid-Ansicht in der normalen Grid-Ansicht angezeigt.

Bearbeitung von Tabellendaten
Sie können Tabellendaten auf folgende Art bearbeiten:

- Spalten mit Drag-and-Drop der Spaltenüberschrift verschieben.
- Spaltendaten für Text-Node mit Hilfe des Menübefehls **XML | Tabelle | Aufsteigend sortieren** (auch absteigend sortieren) sortieren.

- Zeilen mit Hilfe des Menübefehls **XML | Tabelle | Zeile einfügen** (auch Zeile anhängen) einfügen (oder anhängen).

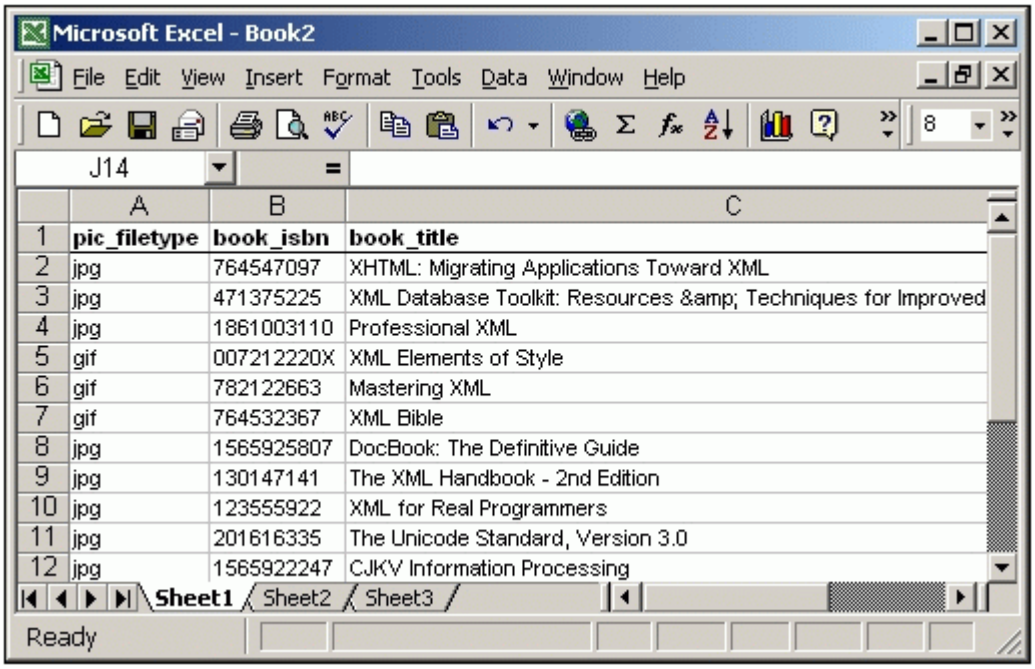
Verschieben von Daten zwischen der Tabellenansicht und externen Applikationen
Die Tabellenstruktur der Tabellenansicht eignet sich auch dazu, Daten aus einem Tabellenkalkulationsprogramm zu übernehmen und umgekehrt. Um Daten aus der Tabellenansicht in eine andere Applikation zu kopieren, wählen Sie die gewünschten Nodes in der Tabelle aus und kopieren Sie die Daten mit Hilfe des Befehls Als strukturierten Text kopieren direkt in z.B. ein Excel Sheet. (Sie können Nodes in der Tabellenansicht durch Klicken auf die Zelle selbst, auf die Spalten- oder Zeilenüberschrift (siehe unten) oder die gesamte Tabelle auswählen. Wenn Sie die gesamte Tabelle oder Spaltenüberschriften auswählen, wird auch der Text der Spaltenüberschriften ausgewählt; ansonsten nicht)



	= pic_filetype	{} book_isbn	{} book_title
1	jpg	0764547097	XHTML: Migrating Applications Toward
2	jpg	0471375225	XML Database Toolkit: Resources &am Development
3	jpg	1861003110	Professional XML
4	gif	007212220X	XML Elements of Style
5	gif	0782122663	Mastering XML
6	gif	0764532367	XML Bible
7	jpg	1565925807	DocBook: The Definitive Guide
8	jpg	0130147141	The XML Handbook - 2nd Edition
9	jpg	0123555922	XML for Real Programmers
10	jpg	0201616335	The Unicode Standard, Version 3.0
11	jpg	1565922247	CJKV Information Processing

Im oben gezeigten Screenshot sehen Sie 11 Buch-Elemente die in der Tabellenansicht in Tabellenform angezeigt werden. Um die gesamte Tabelle in ein Excel Sheet zu kopieren, müssten Sie das Element `book(11)` auswählen, es als strukturierten Text kopieren und in ein Excel Sheet einfügen. Wenn Sie Zelle A1 auswählen und die Tabelle einfügen, sieht das Ergebnis aus, wie unten gezeigt.

© 2007 Altova GmbH



	A	B	C
1	**pic_filetype**	**book_isbn**	**book_title**
2	jpg	764547097	XHTML: Migrating Applications Toward XML
3	jpg	471375225	XML Database Toolkit: Resources & Techniques for Improved
4	jpg	1861003110	Professional XML
5	gif	007212220X	XML Elements of Style
6	gif	782122663	Mastering XML
7	gif	764532367	XML Bible
8	jpg	1565925807	DocBook: The Definitive Guide
9	jpg	130147141	The XML Handbook - 2nd Edition
10	jpg	123555922	XML for Real Programmers
11	jpg	201616335	The Unicode Standard, Version 3.0
12	jpg	1565922247	CJKV Information Processing

Der Datenaustausch funktioniert in beide Richtungen. Sie können daher auch Daten aus jedem Tabellenkalkulationsprogramm kopieren und in der Tabellenansicht direkt in eine Tabelle einfügen. Gehen Sie dabei folgendermaßen vor:

1. Wählen Sie in der externen Applikation einen Bereich aus und kopieren Sie ihn (in die Zwischenablage - in Windows-Systemen mit **Strg+C**).
2. Wählen Sie in Ihrem XML-Dokument in der Tabellenansicht eine einzelne Zelle aus.
3. Fügen Sie die kopierten Daten mit **Strg+V** ein.

Die Daten werden in XMLSpy in die Tabelle eingefügt. Die Tabellenstruktur der Originaltabelle bleibt dabei erhalten und die Tabelle wird beginnend in der Zelle eingefügt, die in der Tabellenansicht ausgewählt ist. Beachten Sie bitte Folgendes:

- Wenn diese Zellen in der Tabellenansicht bereits Daten enthalten, werden die alten Daten durch die neuen überschrieben.
- Wenn für die neuen Daten mehr Zellen benötigt werden, werden diese erstellt.
- Werden jedoch mehr Spalten für die neuen Daten benötigt, werden diese **nicht** erstellt.
- Die Daten in den Zellen werden Inhalt der Elemente, für die die entsprechenden Zellen stehen.

Für komplexere Datenaustauschaufgaben bietet XMLSpy außerdem einige einzigartige Konvertierungsfunktionen zum direkten Importieren oder Exportieren von XML-Daten aus jeder beliebigen Textdatei, jedem Word-Dokument oder jeder Datenbank.

5 Schema/WSDL Design-Ansicht

Die **Schema/WSDL Design-Ansicht** ist bei XML-Schema- und WSDL-Dokumenten aktiv. Die Ansicht ändert sich, je nachdem, ob ein XML-Schema oder ein WSDL-Dokument aktiv ist und wird Schema Design-Ansicht bzw. WSDL Design-Ansicht genannt. Diese Ansichten bieten eine grafische Darstellung des XML-Schema- bzw. WSDL-Dokuments und gestatten Ihnen, XML-Schemas und WSLD-Dokumente auf intuitive Art zu erstellen und zu bearbeiten. Wenn Sie eine lizenzierte Version von SchemaAgent auf Ihrem Rechner installiert haben, haben Sie beim Erstellen von Schemas nicht nur Zugriff auf die Komponenten des derzeit in XMLSpy aktiven Schemas, sondern auch auf die Komponenten vieler anderer Schemas, die in anderen Repositories auf Ihrem Netzwerk gespeichert sind. Im Abschnitt Arbeiten mit SchemaAgent wird beschrieben, wie Sie SchemaAgent installieren und damit arbeiten.

Diese Ansichten sind in den Unterkapiteln dieses Abschnitts ausführlich beschrieben. Beispiele, wie man XML-Schemas und WSDL-Dokumente in der Schema Design-Ansicht bzw. der WSDL Design-Ansicht erstellt, finden Sie im XML-Schema Tutorial und dem WSDL Tutorial.

© 2007 Altova GmbH

5.1 Schema Design-Ansicht

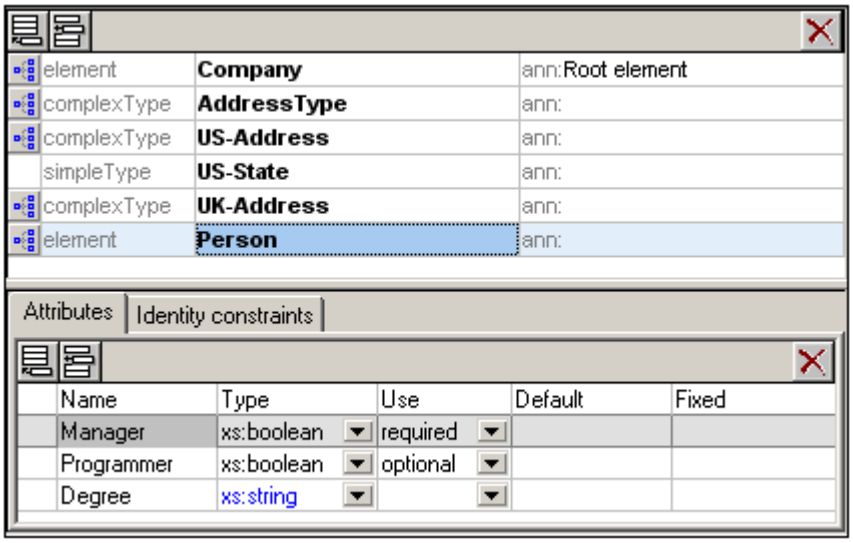
Die Schema Design-Ansicht selbst verfügt über zwei Arten von Ansichten:

- einer Schema-**Hauptübersicht**, in der alle globalen Komponenten (globale Elemente, complexTypes usw.) in einer einfachen Tabelle dargestellt werden.
- Ansichten der Content Models der einzelnen globalen Komponenten (**Content Model-Ansicht**).

Weiter unten erhalten Sie eine kurze Übersicht über die Schema-Übersicht und die Content Model-Ansicht, gefolgt von einer kurzen Beschreibung der in der Schema Design-Ansicht verfügbaren Eingabehilfen. Die beiden Unterkapitel dieses Abschnitts enthalten eine ausführliche Beschreibung der Schema-Übersicht und Content Model-Ansicht.

Bitte beachten Sie: Mit dem SchemaAgent-Feature von Altova, das Sie über XMLSpy aufrufen, können Sie Komponenten aus anderen XML-Schemas direkt verwenden.

Schema-Übersicht
In der Schema-Übersicht sehen Sie eine Liste aller globalen Komponenten des Schemas (globale Elemente, complexTypes, usw).



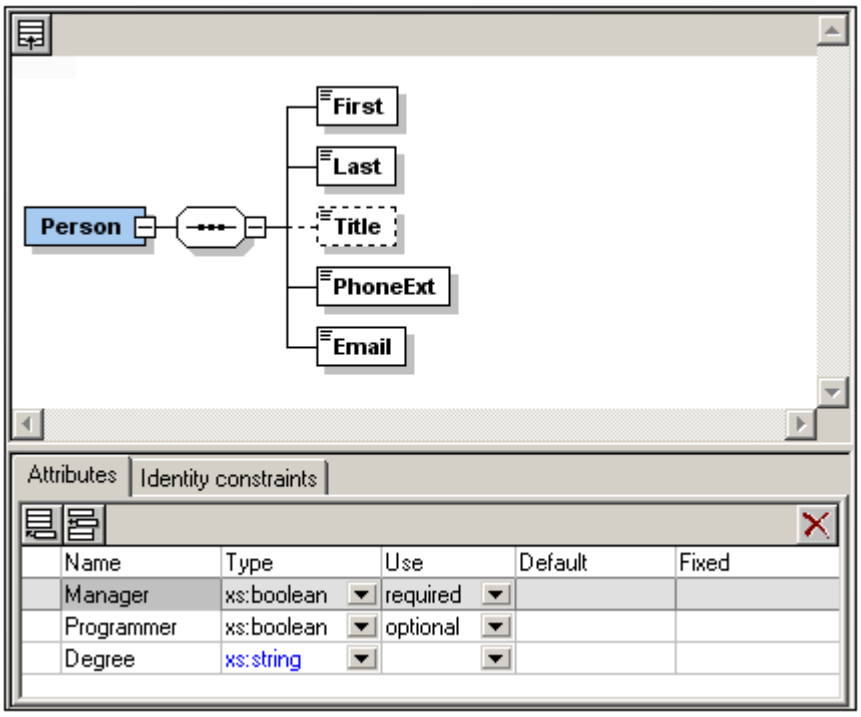
Sie können globale Komponenten einfügen, anhängen oder löschen und ihre Eigenschaften bearbeiten. Verwenden Sie zum Einfügen, Anhängen oder Löschen die entsprechenden Schaltflächen am oberen Rand der Schema-Übersicht. Um Eigenschaften zu bearbeiten, wählen Sie die gewünschte Komponente in der Schema-Übersichtsliste aus und editieren Sie die Eigenschaften entweder über die Eingabehilfen (im rechten Bereich der Ansicht) oder das Attribute-/Identity Constraints- Fenster (im unteren Bereich der Ansicht).

In der Schema-Übersicht stehen Ihnen die folgenden Editierfunktionen zur Verfügung:

- Sie können Komponenten in der Schema-Übersichtsliste mit Drag & Drop verschieben.
- Sie können mit Hilfe der Pfeiltasten navigieren.
- Sie können globale Komponenten, Attribute und Identity Constraints mit Hilfe der Funktionen Ausschneiden/Kopieren-und-Einfügen an eine andere Stelle verschieben und oder von einem Schema in ein anderes verschieben bzw. kopieren.
- Durch Rechtsklick auf eine Komponente öffnen Sie ein Kontextmenü, über das Sie die Annotationsdaten dieser Komponente ausschneiden, kopieren, einfügen oder bearbeiten können.

Content Model-Ansicht
Ein Content Model ist eine Beschreibung der Struktur und des Inhalts eines Elements. Globale Komponenten, die ein Content Model haben können (z.B. Elemente, complexTypes und Model Groups; nicht aber z.b. simpleTypes) werden in der Schema-Übersichtsliste mit einem Symbol links vom Komponentennamen gekennzeichnet. Bei Doppelklick auf dieses Symbol wird das Content Model für diese globale Komponente geöffnet. Alternativ dazu (i) wählen Sie eine Komponente aus und anschließend die Menüoption **Schema-Design | Diagramm anzeigen** oder (ii) doppelklicken Sie im Komponentennavigator(d.h. der Eingabehilfe rechts oben) auf einen Komponentennamen. Beachten Sie, dass immer nur jeweils ein Content Model im Schema geöffnet sein kann. Wenn ein Content Model geöffnet ist, können Sie zum Content Model einer Komponente innerhalb des aktuellen Content Model springen, indem Sie bei gedrückter **Strg**-Taste auf die gewünschte Komponente doppelklicken.

Das Content Model wird in der Content Model-Ansicht als Baumdiagramm (siehe Abbildung unten) dargestellt. Sie können die Darstellungsweise des Diagramms im Dialogfeld "Schema-Ansicht konfigurieren" (Menüeintrag **Schema-Design | Ansicht konfigurieren** einstellen).



Die Content Model-Ansicht verfügt über die folgenden Editierfeatures:

- Jede Ebene (Elemente oder Elementgruppen) im Baumdiagramm ist mittels eines Kompositors mit benachbarten Ebenen verbunden.
- Sie können Objekte des Baumdiagramms (Kompositoren, Elemente, Elementgruppen) mittels Drag-and-Drop verschieben.
- Sie können Objekte aus der Baumstruktur mit Hilfe der Tastenkombinationen Strg + C und Strg + V kopieren bzw. einfügen
- Objekte (Kompositoren, Elemente und Elementgruppen) können über das Kontextmenü (wird durch Rechtsklick auf ein Objekt aufgerufen) hinzugefügt werden.
- Sie können die Eigenschaften eines Objekts in der Details-Eingabehilfe (Kompositoren, Elemente, Elementgruppen) und im Attribute-/Identity Constraints-Fenster bearbeiten.
- Die Attribute und Identity Constraints einer Komponente werden in einem Fenster am

© 2007 Altova GmbH

unteren Rand des Hauptfensters angezeigt. Attribute und Identity Constraints können auch im Content Model-Diagramm anstatt des Attribute/Identity Constraints-Fensters angezeigt werden. Diese Ansichtsoption lässt sich im Dialogfeld Schema-Ansicht konfigurieren einstellen.

Diese Funktionen werden in den Unterkapiteln dieses Abschnitts und im Tutorial näher erläutert.
Um zur Schema-Übersicht zurückzukehren, klicken Sie auf das Symbol **Globale Elemente anzeigen** oder wählen Sie die Menüoption **Schema-Design | Globale Elemente anzeigen** .

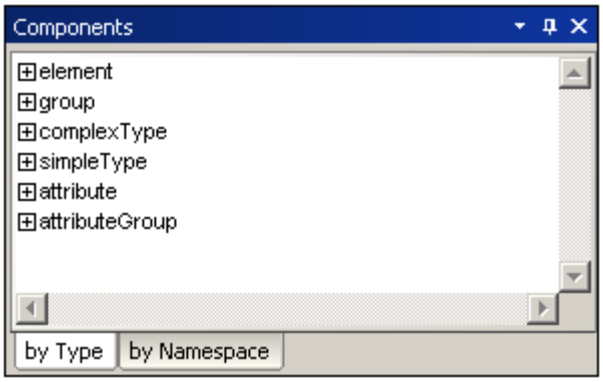
Eingabehilfen in der Schema-Ansicht

Es gibt drei Eingabehilfen in der Schema/WSDL Design-Ansicht: den Komponenten-Navigator, die Details-Eingabehilfe und die Facets-Eingabehilfe. Im Folgenden sind diese näher beschrieben.

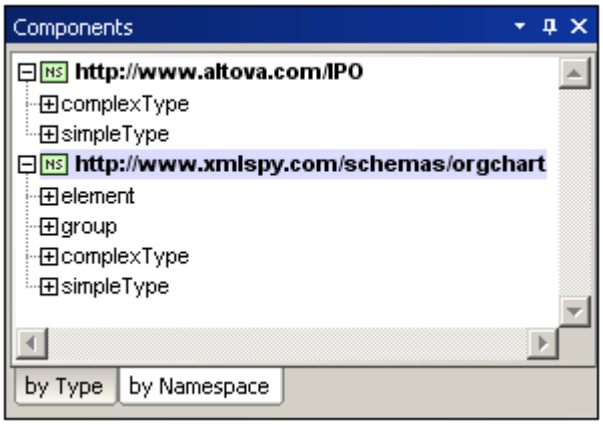
Komponenten-Navigator

Der Komponenten-Navigator ist eine Eingabehilfe in der **Schema/WSDL Design-Ansicht**. Er hat zwei Aufgaben:

- Globale Komponenten in einem Baumdiagramm nach Komponententyp und Namespace (*siehe Abbildung unten)* zu strukturieren. Dadurch steht Ihnen eine strukturierte Ansicht aller globalen Komponenten zur Verfügung.
- Ihnen das Navigieren zur Content Model-Ansicht einer globalen Komponenten zu ermöglichen und diese anzuzeigen - falls die Komponente ein Content Model hat. Wenn eine Komponente kein Content Model hat, wird die Komponente in der Schema-Ansicht markiert. Auch aus anderen Schemas inkludierte oder importierte globale Komponenten werden im Komponenten-Navigator angezeigt.



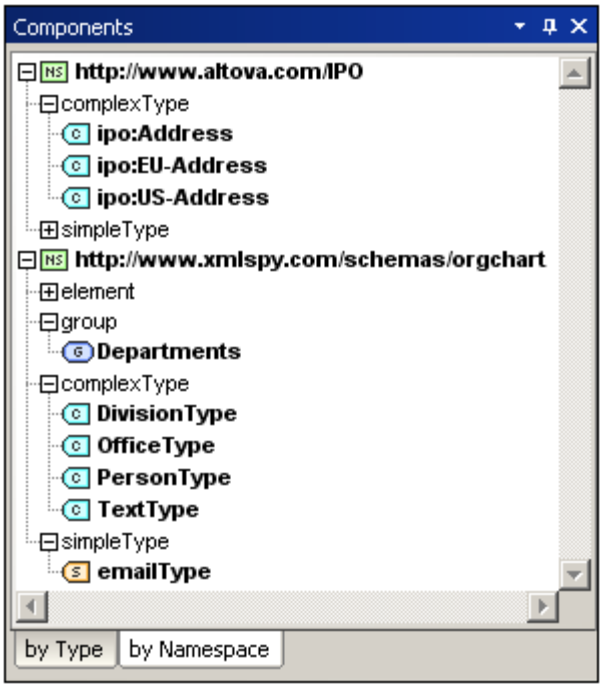
Auf dem Register "by Type" (*siehe oben)* werden die globalen Komponenten nach ihrem Komponententyp in einer Baumstruktur gegliedert. Auf dem Register Namespace (*siehe unten)* sind die Komponenten zuerst nach Namespace und anschließend nach Komponententyp geordnet. Beachten Sie, dass ein Komponententyp nur dann in einer Baumstruktur angezeigt wird, wenn mindestens eine Komponente dieses Typs im Schema vorhanden ist.



In der Baumstruktur werden globale Komponenten in die folgenden sechs Gruppen eingeordnet:

- Element-Deklarationen (Elemente)
- Model Groups (Groups)
- ComplexTypes
- SimpleTypes
- Attribut-Deklarationen (Attribute)
- Attribut-Gruppen

Wenn Sie eine Komponenten-Typ-Gruppe in der Baumstruktur erweitern, werden alle Komponenten in dieser Gruppe angezeigt (*siehe Abbildung unten)*. Auf diese Art können Sie schnell zur gewünschten Komponente navigieren.



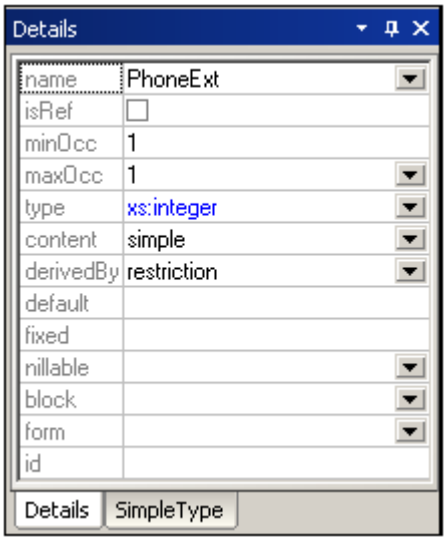
Wenn eine Komponente ein Content Model hat (d.h. wenn es sich um ein Element, eine Group oder einen complexType handelt), wird mit einem Doppelklick darauf das Content Model dieser Komponente in der Content Model-Ansicht (im Hauptfenster) angezeigt. Wenn die Komponente kein Content Model hat (d.h. wenn es sich um einen simpleType, ein Attribut oder eine

Attribut-Gruppe handelt), so wird die Komponente in der Schema-Übersicht (im Hauptfenster) markiert.

Bitte beachten Sie: Wenn es sich bei der Komponente um ein inkludiertes oder importiertes Schema handelt, wird dieses geöffnet (wenn es nicht bereits offen ist) und es wird entweder das Content Model der Komponente in der Content Model-Ansicht angezeigt oder die Komponente wird in der Schema-Übersicht markiert.

Details-Eingabehilfe

Die Details-Eingabehilfe steht in der **Schema/WSDL Design-Ansicht** zur Verfügung. Sie enthält editierbare Informationen über den im Hauptfenster ausgewählten Kompositor/die ausgewählte Komponente. Wenn Sie eine Schemadatei, die Datenbank-Extensions enthält, bearbeiten, wird unter Umständen ein zusätzliches Register mit Informationen über die DB Extensions angezeigt. Derzeit werden Oracle-, SQL Server- und Tamino-Datenbanken unterstützt.

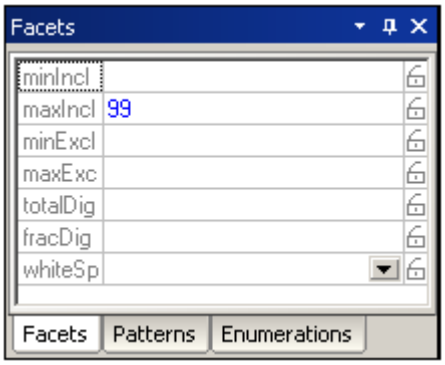


Um die Eigenschaften des derzeit ausgewählten Kompositors oder der ausgewählten Komponente zu ändern, doppelklicken Sie auf das zu editierende Feld und bearbeiten Sie es oder geben Sie Text direkt ein. Wenn in dem gewünschten Feld eine Dropdown-Liste zur Verfügung steht, wählen sie den gewünschten Wert aus; dieser Wert wird in dem Feld übernommen.

Änderungen, die Sie über die Details-Eingabehilfe vornehmen, werden sofort im Content Model-Diagramm angezeigt.

Facets-Eingabehilfe

Die Facets-Eingabehilfe steht in der **Schema/WSDL Design-Ansicht** zur Verfügung und gestattet die Eingabe der Werte von Facets, Patterns und Enumerations. Ein Beispiel zur Verwendung von Facets in einem XML-Schema finden Sie in den entsprechenden Abschnitten des Tutorials.



Um Facets, Patterns oder Enumerations in der Facets-Eingabehilfe zu ändern, gehen Sie vor wie folgt:

1. Wählen Sie das entsprechende Register aus (Facets, Patterns oder Enumerations)
2. Wählen Sie einen Wert aus der Dropdown-Liste aus, falls vorhanden. Doppelklicken Sie andernfalls auf eine Zeile und bearbeiten Sie den Text manuell bzw. geben Sie ihn manuell ein.

Bitte beachten Sie: Sie können die Patterns und Enumerations einer Komponente mit Hilfe der Tastenkürzel zum Ausschneiden, Kopieren und Einfügen (Strg+X, Strg+C bzw. Strg+V) in eine andere Komponente kopieren. Wählen sie in der Facets-Eingabehilfe ein oder mehrere Patterns oder Enumerations aus, um diese auszuschneiden oder zu kopieren, klicken Sie anschließend in die Facets-Eingabehilfe der Zielkomponente und fügen Sie sie ein.

5.1.1 Schema-Übersicht

Auf der obersten Ebene eines XML-Dokuments (d.h. auf Ebene der Child-Elemente des `schema`-Elements) können die folgenden fünf Basiskomponenten definiert werden:

- Annotation
- Typ-Definition (simple oder complex)
- Deklaration (Element oder Attribut)
- Attribute Group
- Model Group

Diese Komponenten auf der obersten Ebene werden **globale Komponenten** genannt. In der **Schema-Übersicht** werden allen globen Komponenten in Ihrem Schema in Tabellenform aufgelistet. Einige globale Komponenten (wie z.B. complexTypes, Element-Deklarationen und Model Groups) können ein Content Model haben, das Struktur und Inhalt der Komponente beschreibt. Andere globale Komponenten (wie z.B. Annotationen, simpleTypes und Attribut-Gruppen) haben kein Content Model. Komponenten, die ein Content Model haben können, sind durch ein Symbol links vom Komponentennamen gekennzeichnet. Wenn Sie auf dieses Symbol klicken, wird die Content Model-Ansicht für diese globale Komponente angezeigt.

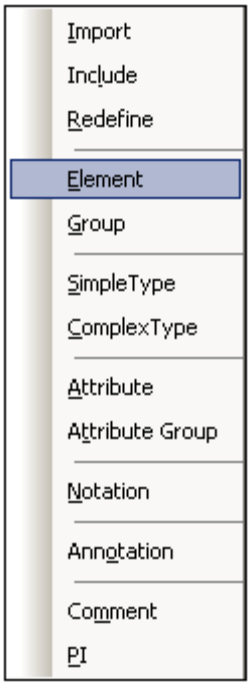
Grundlegende Begriffe

- *SimpleType und complexType.* Ein simpleType dient dazu, alle Attribute und Elemente zu definieren, die nur Text und kein Attribut enthalten. Ein simpleType hat daher kein Content Model, sondern nur Text (der durch den Datentyp eingeschränkt werden kann). Ein complexType hat dagegen mindestens ein Child-Element oder -Attribut. Wenn Sie ein Child-Element für ein Element deklarieren, wird diesem automatisch der Typ complexType zugewiesen.

- *Globale und lokale Komponenten.* Bei einer globalen Komponente kann es sich um jede der fünf oben angeführten Arten handeln. Eine globale Komponente kann in der Schema-Übersicht definiert werden und scheint daraufhin sofort in der Liste der globalen Komponenten in der Schema-Übersicht auf. Wenn es sich bei der globalen Komponente um einen complexType, eine Element-Deklaration oder eine Model Group handelt, können Sie anschließend ihr Content Model definieren, indem Sie es in der Content Model-Ansicht bearbeiten. Sobald eine globale Komponente definiert ist, kann sie von lokalen Komponenten referenziert werden. Eine lokale Komponente wird direkt im Content Model einer Komponente erstellt. Beachten Sie, dass eine lokale Komponente in der Content Model-Ansicht (über das Kontextmenü) in eine globale Komponente konvertiert werden kann.

Erstellen globaler Komponenten
So erstellen Sie in der Schema-Übersicht eine globale Komponente:

1. Klicken Sie oben in der Schema-Übersicht auf das Symbol "Einfügen" oder "Anhängen" . Daraufhin wird ein Menü angezeigt, aus dem Sie verschiedene Komponententypen (Element, simple type, complex type, model group usw.) auswählen können.



2. Wählen Sie den gewünschten Komponententyp aus. In der Liste der globalen Komponenten wird ein Eintrag dieses Typs erstellt.
3. Geben Sie den Namen der Komponente hier ein und drücken Sie die **Eingabetaste**. Der Name der neuen globalen Komponente wird zur entsprechenden Liste bzw. zu den entsprechenden Listen (Elm, Grp, Com, Sim, usw.) in der Komponenten-Navigator-Eingabehilfe hinzugefügt. Sie können das Content Model der neuen globalen Komponente entweder durch Doppelklicken auf den Komponentennamen im Komponenten-Navigator oder durch Klicken auf das Symbol links vom Komponentennamen in der Liste der globalen Komponenten bearbeiten.

Bitte beachten Sie:

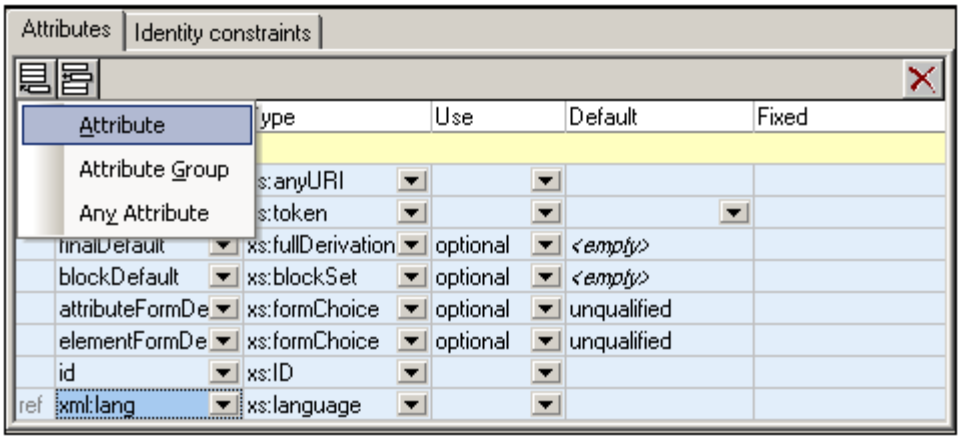
- Sie können außerdem eine globale Komponente erstellen, während Sie in der Content Model-Ansicht arbeiten. Klicken Sie mit der rechten Maustaste auf eine beliebige Stelle im Fenster und wählen Sie **Neues Global | Element**.
- Beim Bearbeiten in der Content Model-Ansicht können Sie ein lokales Element zu einem globalen - oder sogar zu einem complexType machen, wenn das Element ein Child-Element oder -Attribut hat. Wählen Sie das lokale Element aus, rechtsklicken Sie in das Fenster und wählen Sie **Als global definieren | Element** oder **Als global definieren | Complex Type**.

Löschen globaler Komponenten
So löschen Sie eine globale Komponente:

1. Wählen Sie die globale Komponente in der Schema-Übersicht in der Liste der globalen Komponenten aus.
2. Drücken Sie die **Entfernen**-Taste oder klicken Sie am oberen Rand der Schema-Übersicht auf das Symbol **Löschen** .

Attribute und Identity Constraints von Komponenten
Sie können Attribute und Identity Constraints für Komponenten entweder in der Schema-Übersicht oder in der Content Model-Ansicht definieren. In der Schema-Übersicht werden die Attribute und Identity Constraints einer Komponente im Attribute-/Identity Constraints-Fenster im unteren Bereich des Schema-Übersichtsfensters angezeigt und können hier editiert werden. In der Content Model-Ansicht können Attribute und Identity Constraints entweder im Attribute-/Identity Constraints-Fenster am unteren Rand des Content Model-Fensters oder innerhalb des Content Model-Diagramms selbst angezeigt werden, wo Sie diese bearbeiten können. Über eine Einstellung im Dialogfeld Schema-Ansicht konfigurieren, die Sie über den Menübefehl **Schema-Design | Ansicht konfigurieren** aufrufen, können Sie einstellen, wie Attribute und Identity Constraints in der Content Model-Ansicht angezeigt werden sollen.

Definieren von Attributen für eine Komponente
Attribute für eine Komponente lassen sich über das Attribute-/Identity Constraints-Fenster am unteren Rand des Schema-Übersichtsfensters definieren.



So definieren Sie Attribute für eine globale Komponente, für die Attribute zulässig sind:

1. Wählen Sie die globale Komponente in der Liste der globalen Komponenten aus.
2. Wählen Sie im Attribute-/Identity Constraints-Fenster das Register "Attribute".

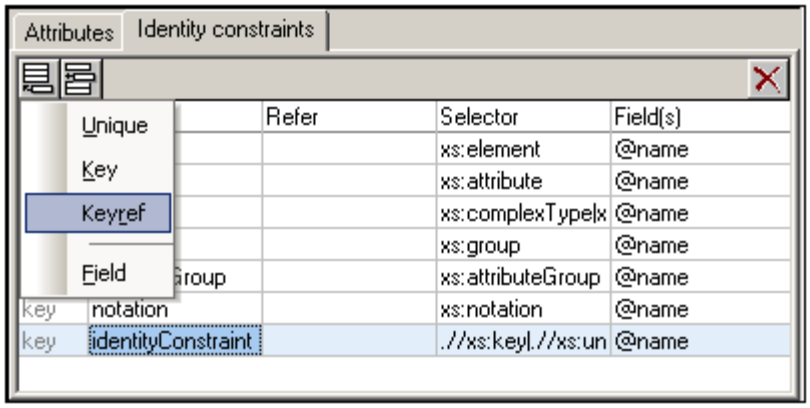
 © 2007 Altova GmbH

3. Klicken Sie auf das Symbol "Anhängen" oder "Einfügen" links oben auf dem Register "Attribute".
4. Wählen Sie im Popup-Menü, das daraufhin angezeigt wird, den gewünschten Attributtyp aus. In der Attributliste wird ein Eintrag erstellt.
5. Geben Sie in dem soeben erstellten Eintrag die Eigenschaften des Attributs an.

Bitte beachten Sie: Sie können Attribute für globale Komponenten auch in der Content Model-Ansicht definieren. Wählen sie die globale Komponente aus und definieren Sie anschließend Attribute wie oben beschrieben.

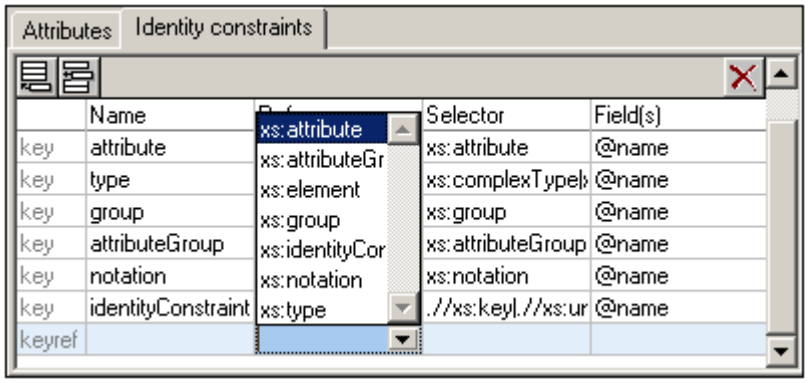
Definieren von Identity Constraints für eine Komponente
Zum Definieren von Identity Constraints für eine Komponente wird das Attribute-/Identity Constraints-Fenster im unteren Bereich des Schema-Übersichtsfensters verwendet. So definieren Sie Identity Constraints für eine Komponente:

1. Wählen Sie die globale Komponente in der Liste der globalen Komponenten aus.
2. Wählen Sie im Attribute-/Identity Constraints-Fenster das Register "Identity Constraints" aus.
3. Klicken Sie auf das Symbol "Anhängen" oder "Einfügen" in der linken oberen Ecke des Registers.
4. Wählen Sie im Popup-Fenster die Art des gewünschten Identity Constraint aus.



In der Liste der Identity Constraints wird ein Eintrag erstellt.

5. Geben Sie in diesen Eintrag die Eigenschaften des Identity Constraint ein. `Selector` gibt den Nodeset an, für den der Identity Constraint gilt. `Field` identifiziert den Wert, der innerhalb des von der `Selector`-Eigenschaft identifizierten Nodesets eindeutig sein muss. Wenn Sie als Identity Constraint Typ `Keyref` ausgewählt haben, ist die `Refer`-Eigenschaft aktiviert und in der Dropdown-Liste werden alle im Schema definierten Schlüssel aufgelistet (*siehe Abbildung unten*).



Bitte beachten Sie: Sie können auch für globale Komponenten in der Content Model-Ansicht Identity Constraints definieren. Wählen Sie die globale Komponente aus und definieren Sie wie oben beschrieben Identity Constraints.

5.1.2 Content Model-Ansicht

In der **Content Model-Ansicht** können Sie das Content Model der folgenden drei Komponententypen grafisch mit wenigen Mausklicks definieren:

- ComplexTypes,
- Element-Deklarationen und
- Model Goups

Alle anderen Schema-Komponenten (Annotationen, Attribut-Deklarationen, simpleTypes usw.) haben kein Content Model.

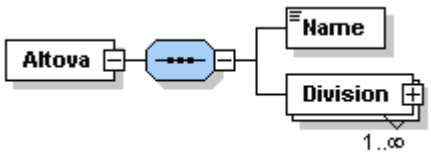
In der Content Model-Ansicht werden die verschiedenen Teile des Content Model grafisch dargestellt. Diese Teile lassen sich in zwei große Gruppen unterteilen: **Kompositoren** und **Komponenten**. Normalerweise wird ein Kompositor hinzugefügt und anschließend die gewünschten Child-Komponenten.

Kompositoren
Ein **Kompositor** definiert die Reihenfolge, in der Child-Elemente angeordnet sind. Es gibt drei Kompositoren: `sequence`, `choice` und `all`. So fügen Sie einen Kompositor ein:

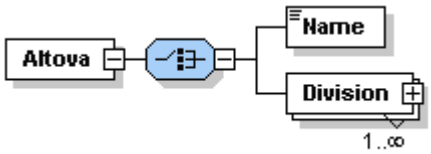
1. Rechtsklicken Sie auf das Element, zu dem Sie Child-Elemente hinzufügen möchten.
2. Wählen Sie den Befehl **Child hinzufügen | Sequence** (oder **Choice** oder **All**).

Daraufhin wird ein Kompositor hinzugefügt, der folgendermaßen aussieht:

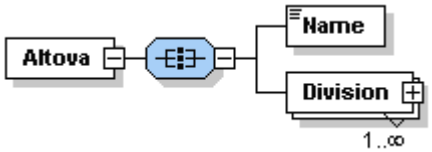
- **Sequence**



- **Choice**



- **All**



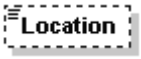
Um den Kompositor zu ändern, rechtsklicken Sie darauf und wählen Sie **Modell ändern | Sequence** (oder **Choice** oder **All**). Nachdem Sie den Kompositor hinzugefügt haben, müssen Sie (ein) Child-Element(e) oder eine Model Group hinzufügen.

Komponenten im Content Model
Unten finden Sie eine Liste von Komponenten, die in Content Models verwendet werden. Die grafische Darstellung der einzelnen Komponenten gibt Ihnen genaue Informationen über die Art der Komponente und ihre strukturellen Eigenschaften.

- Obligatorisches Einzelelement

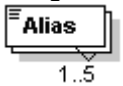
Details: Das Rechteck zeigt an, dass es sich um ein Element handelt, die durchgezogene Umrandung bedeutet, dass das Element "required" also obligatorisch ist. Das Fehlen eines Zahlenbereichs weist darauf hin, dass es ein Einzelelement ist (d.h. `minOcc=1` und `maxOcc=1`). Der Name des Elements ist `Country`. Die blaue Farbe zeigt an, dass das Element gerade ausgewählt ist (eine Komponente wird durch Anklicken ausgewählt). Wenn eine Komponente nicht ausgewählt ist, ist sie weiß.

- Optionales Einzelelement



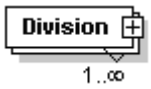
Details: Rechteck = Element; gestrichelte Umrandung = optional; fehlender Zahlenbereich = `minOcc=0` und `maxOcc=1`. Der Element-Name ist `Location`.
Bitte beachten Sie: Im Kontextmenü können Sie ein obligatorisches Element über die Menüoption **Optional** in ein optionales umwandeln.

- Obligatorisches mehrmals vorkommendes Element

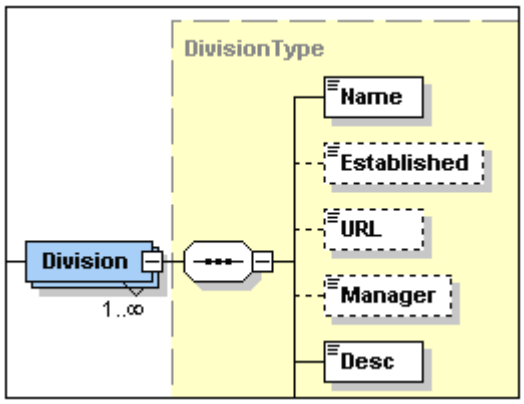


Details: Rechteck = Element; durchgezogene Umrandung = obligatorisch; Zahlenbereich 1..5 = `minOcc=1` und `maxOcc=5`. Der Element-Name ist `Alias`.

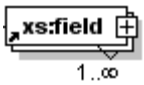
- Obligatorisches mehrfach vorkommendes Element, das Child-Elemente enthält



Details: Rechteck = Element; durchgezogene Umrandung = obligatorisch; Zahlenbereich 1..unendlich = `minOcc=1` und `maxOcc=unbounded`; Plus-Zeichen = complex content (d.h. es hat mindestens ein Child-Element oder -Attribut). Der Element-Name ist `Division`.
Bitte beachten Sie: Über die Kontext-Menüoption **Unbounded** können Sie `maxOcc` in `unbounded` ändern.
Wenn Sie auf das +-Zeichen des Elements klicken, wird die Baumstruktur erweitert und Sie sehen die Child-Elemente.

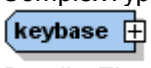


- Element, das ein globales Element referenziert



Details: Pfeil in der linken unteren Ecke = Element, das ein globales Element referenziert; Rechteck = Element; durchgezogene Umrandung = obligatorisch; Zahlenbereich 1..unendlich = `minOcc=1` und `maxOcc=unbounded`; Plus-Zeichen = complex content (d.h. mindestens ein Child-Element oder -Attribut). Der Element-Name ist `xs:field`.
Bitte beachten Sie: Ein globales Element kann von simple- und complex type-Definitionen aus referenziert werden, sodass Sie eine globale Deklaration an mehreren Stellen in Ihrem Schema wiederverwenden können. Es gibt zwei Möglichkeiten, um eine Referenz auf ein globales Element zu erstellen: (i) durch Eingabe eines Namens für das lokale Element, der mit dem des globalen Elements identisch ist; und (ii) durch Rechtsklick auf das lokale Element und Auswahl der Kontextmenüoption **Reference**. Um die Definition eines globalen Elements anzusehen, halten Sie die **Strg**-Taste gedrückt und doppelklicken Sie auf das Element. Alternativ dazu können Sie auch rechtsklicken und den Befehl **Gehe zu Definition** auswählen. Wenn Sie ein Element referenzieren, das nicht existiert, wird der Elementname rot angezeigt. Dies ist eine Warnung, dass es keine referenzierbare Definition gibt.

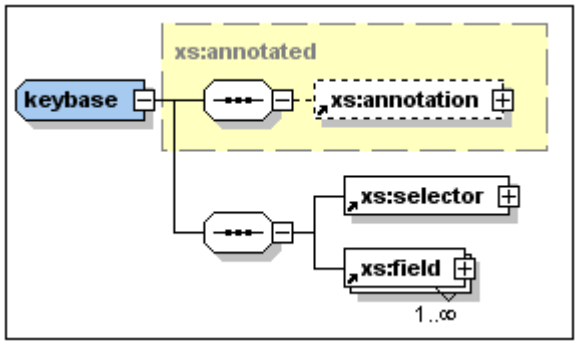
- ComplexType



Details: Ein unregelmäßiges Sechseck mit einem Plus-Zeichen (siehe oben) kennzeichnet einen complexType. Der hier dargestellte complexType hat den Namen `keybase`. Dieses Symbol bedeutet, dass es sich um einen globalen complexType handelt. Ein globaler complexType wird in der Schema-Übersicht deklariert. Sein Content Model wird normalerweise in der Content Model-Ansicht definiert. Ein globaler

 © 2007 Altova GmbH

complexType kann für folgende Zwecke verwendet werden: (i) als Datentyp eines Elements oder (ii) als Base Type eines anderen complexType, indem Sie ihn in der Details-Eingabehilfe (entweder in der Content Model-Ansicht oder in der Schema-Übersicht) dem Element bzw. complexType zuweisen.



Der oben dargestellte complexType `keybase` wurde in der Schema-Übersicht mit einem Base Type `xs:annotated` deklariert. Der Base Type wird als Rechteck mit einer grauen gestrichelten Umrandung und gelbem Hintergrund dargestellt. Anschließend wurden in der Content Model-Ansicht die Child-Elemente `xs:selector` und `xs:field` erstellt. (Beachten Sie die kleinen Pfeile in der linken unteren Ecke der Rechtecke mit der Beschriftung `xs:selector` und `xs:field`. Diese Pfeile bedeuten, dass beide Elemente globale Elemente dieses Namens referenzieren.)

Ein lokaler complexType wird direkt in der Content Model-Ansicht durch Erstellung eines Child-Elements oder -Attributs für ein Element definiert. Es gibt kein separates Symbol für lokale complexTypes.

Bitte beachten Sie: Der Base Type eines Content Models wird als Rechteck mit grauer gestrichelter Umrandung und gelber Hintergrundfarbe dargestellt. Um das Content Model des Base Type aufzurufen, doppelklicken Sie auf seinen Namen.

- Model Group

  Details: Das unregelmäßige Achteck mit einem Plus-Zeichen (siehe Abbildung oben) kennzeichnet eine Model Group. Eine Model Group gestattet Ihnen, Element-Deklarationen zu definieren und wiederzuverwenden.
  Bitte beachten Sie: Wenn die Model Group (in der Schema-Übersicht) deklariert wird, erhält sie einen Namen. Anschließend wird (in der Content Model-Ansicht) der Content Mode durch Zuweisung eines Child-Kompositors, der die Element-Deklarationen enthält, definiert. Wenn die Model Group verwendet wird, wird Sie (in der Content Model-Ansicht) innerhalb des Content Models einer anderen Komponente als Child eingefügt oder angehängt.

- Wildcards

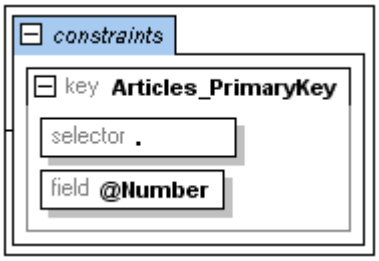
  Details: Das unregelmäßige Achteck mit `any` auf der linken Seite kennzeichnet eine Wildcard.
  Bitte beachten Sie: Wildcards werden als Platzhalter für Elemente verwendet, die im Schema nicht definiert sind oder aus anderen Namespaces stammen. `##other` = Elemente können zu jedem Namespace gehören, bei dem es sich nicht um den im Schema definierten Target Namespace handelt; `##any` = Elemente können zu jedem

Namespace gehören; `##targetNamespace` = Elemente müssen zum Target Namespace gehören, der im Schema definiert wurde; `##local` = Elemente können nicht zu jedem beliebigen Namespace gehören; *`anyURI`* = Elemente gehören zu dem von Ihnen angegebenen Namespace.

- Attribute

Details: Attribute werden durch das kursiv geschriebene Wort *'attributes'* in einem erweiterbaren Rechteck dargestellt. Jedes Attribut wird in einem Rechteck mit einer strichlierten Umrandung dargestellt.
Bitte beachten Sie: Attribute können in der Details-Eingabehilfe bearbeitet werden. Sie können Attribute im Diagramm der Content Model-Ansicht oder in einem Fenster unterhalb der Content Model-Ansicht anzeigen. Durch Klicken auf das Symbol "Attribute direkt in Diagramm anzeigen" können Sie zwischen den beiden Ansichten hin- und herwechseln. Wenn Attribute im Diagramm der Content Model-Ansicht angezeigt werden, werden alle Attribute eines Einzelelements in einem Rechteck mit gestrichelter Umrandung dargestellt. Um die Reihenfolge von Attributen eines Elements zu ändern, ziehen Sie das Attribut aus dem Kästchen heraus und lassen Sie die Maustaste los, wenn an der gewünschten Stelle ein Pfeil erscheint.

- Identity Constraints



Details: Identity Constraints werden durch das kursiv gedruckte Wort '*constraints*' in einem erweiterbaren Rechteck dargestellt.
Bitte beachten Sie: Die im Content Model einer Komponente aufgelisteten Identity Constraints zeigen Constraints, die mit Hilfe der Elemente `key` und `keyref` definiert wurden und das Element `unique` haben. Identity Constraints, die mit Hilfe des `ID` -Datentyps definiert wurden, werden im Content Model-Diagramm nicht angezeigt, jedoch in der Details-Eingabehilfe. Identity Constraints können in der Content Model-Ansicht oder in einem Fenster unterhalb der Content Model-Ansicht angezeigt werden. Durch Klicken auf das Symbol "Contraints direkt im Diagramm anzeigen" können Sie zwischen den beiden Ansichten hin- und herwechseln.

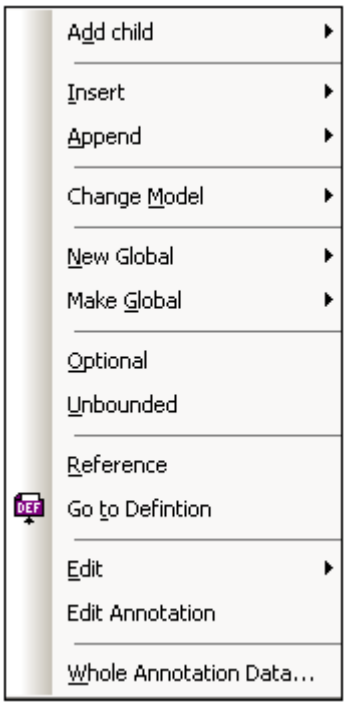
Bitte beachten Sie:

- Durch Klicken auf das Symbol "Vordefinierte Details hinzufügen" können Sie Eigenschaften-Beschreibungen, die Sie im Dialogfeld Schema-Ansicht konfigurieren definiert haben, ein oder ausblenden.
- Durch Auswahl der Symbole "Attribute direkt im Diagramm anzeigen" und

"Constraints direkt im Diagramm anzeigen" können Sie Attribute und Identity Constraints wahlweise entweder im Content Model-Diagramm selbst oder in einem Fenster unterhalb der Content Model-Ansicht anzeigen lassen.

- In der Content Model-Ansicht können Sie zur Content Model-Ansicht einer beliebigen globalen Komponente im aktuellen Content Model springen, indem Sie die **Strg**-Taste gedrückt halten und auf die gewünschte Komponente doppelklicken. Durch Doppelklicken auf den Namen eines Base Type können Sie das Content Model dieses Base Type aufrufen.

Andere Editiermöglichkeiten in der Content Model-Ansicht
Die Editierbefehle für die Content Model-Ansicht können über das Kontextmenü (*siehe Screenshot unten)* aufgerufen werden, das Sie durch einen Rechtsklick in die Content Model-Ansicht öffnen können. Im Folgenden wird beschrieben, welche Funktionen Ihnen zur Verfügung stehen.



Hinzufügen von Child-Kompositoren/Komponenten und Einfügen/Anhängen von Kompositoren/Komponenten

1. Rechtsklicken Sie auf den Kompositor bzw. die Komponente. Daraufhin wird das Kontextmenü (in dem nur die zulässigen Befehle aktiv sind) geöffnet.
2. Wählen Sie den gewünschten Befehl aus.

Ändern eines Kompositors

1. Rechtsklicken Sie auf den gewünschten Kompositor.
2. Wählen Sie die Kontextmenüoption **Modell ändern** und wählen Sie im Untermenü den Kompositor aus, den Sie verwenden möchten. (Der aktuell ausgewählte Kompositor ist durch ein Häkchen markiert.) Wenn ein Kompositor an dieser Stelle nicht verwendet werden darf, ist er ausgegraut.

Erstellen globaler Komponenten

- Um eine globale Komponente zu erstellen, rechtsklicken Sie an eine beliebige Stelle in der Content Model-Ansicht, wählen Sie den Befehl **Neues Global** und aus dem Untermenü die gewünschte Komponente aus.
- Um ein lokales Element zu einem globalen Element oder complexType zu machen, rechtsklicken Sie auf das lokale Element, wählen Sie **Als global definieren** und wählen Sie im Untermenü entweder **Element** oder **Complex type** aus. Wenn eine dieser Komponenten an dieser Stelle nicht erstellt werden kann, ist sie ausgegraut.

Ändern der "Occurrence"-Definition
Sie können zwischen einem "minimum occurrence" und "maximum occurrence" Wert eines Kompositors zwischen `0` und `1` (für `minOccurs`) bzw. `1` und `unbounded` (für `maxOccurs`) wechseln. Gehen Sie dazu vor wie folgt:

1. Rechtsklicken Sie auf den Kompositor oder die Komponente, für die der Occurence-Wert geändert werden muss.
2. Wählen Sie die Kontextmenüoption **Optional**, um den `minOccurs`-Wert auf 1 zu setzen. Wählen Sie die Menüoption **Unbounded,** um den `maxOccurs` Wert auf `unbounded` einzustellen. Deaktivieren Sie die Option **Unbounded**, um `maxOccurs` auf 1 zu setzen. Die jeweils ausgewählte Option ist links neben dem Menüeintrag durch ein Häkchen gekennzeichnet.

Wechseln zwischen einer lokalen und einer globalen Definition
Wenn ein globales Element vorhanden ist, das denselben Namen wie ein lokales Element hat, können Sie die globale Definition referenzieren oder die lokale Definition verwenden. Gehen Sie dabei vor wie folgt:

1. Rechtsklicken Sie auf das Element.
2. Wählen Sie die Kontextmenüoption **Reference**. Wenn das globale Element referenziert wird, wird dieser Menüeintrag mit einem Häkchen versehen. Wenn die lokale Definition verwendet wird, erhält der Menüeintrag **Reference** kein Häkchen.

Zu einer Definition springen
Sie können von einem Content Model zur Definition jeder globalen Komponente springen, die in diesem Content Model enthalten ist. Gehen Sie dazu vor wie folgt:

1. Rechtsklicken Sie auf die globale Komponente. Dabei kann es sich um das gelbe Rechteck eines Base Type, ein Element, das ein globales Element referenziert oder eine Model Group handeln.
2. Wählen Sie die Kontextmenüoption **Gehe zu Definition**. Daraufhin wird die Content Model-Ansicht der globalen Komponente geöffnet.

Alternativ dazu doppelklicken Sie auf den Namen des Base Type oder drücken Sie die Strg-Taste und doppelklicken Sie auf das referenzierende Element oder die Model Group.

Bearbeiten von Elementnamen

1. Rechtsklicken Sie auf das Element.
2. Wählen Sie die Kontextmenüoption **Bearbeiten | Name** und ändern Sie den Namen.

Alternativ dazu können Sie auch auf den Elementnamen doppelklicken und die Änderung eingeben.

 © 2007 Altova GmbH

Erstellen und Bearbeiten von Dokumentation für einen Kompositor oder eine Komponente
Sie können zu einzelnen Kompositoren und Komponenten als Hilfe für Schema-Editoren Dokumentation hinzufügen. Gehen Sie dazu vor wie folgt:

1. Rechtsklicken Sie auf den Kompositor bzw. die Komponente.

2. Wählen Sie die Kontextmenüoption **Annotation in Diagramm bearbeiten**. Daraufhin wird der Raum unterhalb des Kompositors/der Komponente markiert und Sie können beschreibenden Text zum Kompositor bzw. der Komponente eingeben. In der Textansicht wurden dadurch die Elemente `annotation` und `annotation/documentation` erstellt. Das Element `documentation` enthält den von Ihnen eingegebenen beschreibenden Text.

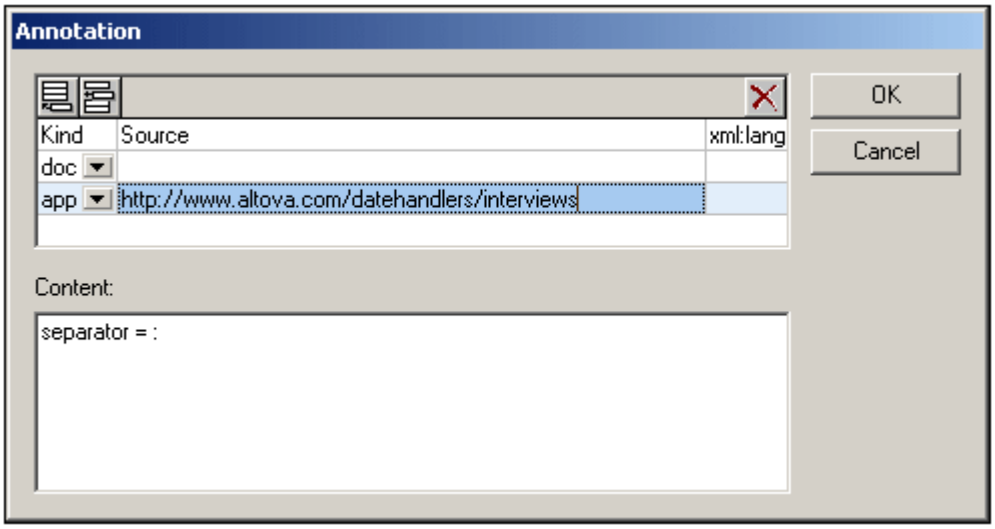
Alternativ dazu können Sie auf den Kompositor/die Komponente rechtsklicken und die Menüoption "Alle Annotationsdaten" wählen. Im Dialogfeld "Annotation", das daraufhin angezeigt wird, können Sie ein Dokumentationselement anhängen oder einfügen und Inhalt dafür eingeben.

Um bereits vorhandenen Dokumentationstext zu bearbeiten, können Sie eine der oben beschriebenen Methoden verwenden, schneller ist es jedoch, wenn Sie im Diagramm auf die Annotation doppelklicken und diese direkt bearbeiten.

Erstellen und Bearbeiten von Applikationsinformationen für einen Kompositor oder eine Komponente

1. Rechtsklicken Sie auf den Kompositor oder die Komponente.

2. Wählen Sie die Kontextmenüoption **Alle Annotationsdaten**. Damit wird das Dialogfeld "Annotation" (*siehe Abbildung unten)* aufgerufen. Wenn für dieses Element eine Annotation (entweder documentation oder appinfo) vorhanden ist, sehen Sie dafür im Dialogfeld eine entsprechende Zeile.



3. Um ein `appinfo`-Element zu erstellen, klicken Sie auf das Symbol "Anhängen" oder "Einfügen" in der linken oberen Ecke, um eine neue Zeile anzuhängen bzw. einzufügen.
4. Wählen Sie im Feld "Kind" der neuen Zeile die Option `app` aus der Dropdown-Liste aus.
5. Geben Sie im Content-Bereich des Dialogfelds das Skript oder die Informationen ein, die von der Applikation verarbeitet werden sollen.
6. Optional dazu können Sie im Feld "Source" eine Quell-URI eingeben, unter der die verarbeitende Applikationen weitere Informationen abrufen kann.

Verwendung von Tastenkürzeln in der Content Model-Ansicht
Sie können Elemente in der Content Model-Ansicht mit den Kürzeln **Strg + C** und **Strg + V** kopieren und einfügen.

So kopieren Sie Elemente und fügen diese ein:
1. Wählen Sie die zu kopierenden Elemente aus. Klicken Sie auf ein Element, um es auszuwählen. Um mehr als ein Element auszuwählen, klicken Sie auf das erste Element und halten Sie die Umschalttaste gedrückt.
2 Drücken Sie **Strg + C.** Die Elemente werden in die Zwischenablage kopiert.
3. Wählen Sie den Kompositor aus, in den Sie die Elemente kopieren möchten.
4. Drücken Sie **Strg + V**. Die Elemente werden als Child-Elemente des Kompositors eingefügt.

So kopieren Sie Elemente und fügen Sie in umgekehrter Reihenfolge ein:
1. Klicken Sie auf das unterste Element, das Sie kopieren möchten und wählen Sie weitere Elemente mit Hilfe der Umschalttaste und des nach-oben-Pfeils aus.
2. Drücken Sie **Strg + C**. Die Elemente werden in die Zwischenablage kopiert.
3. Wählen Sie den Kompositor aus, in den Sie die Elemente kopieren möchten.
4. Drücken Sie **Strg + V**. Die Elemente werden so eingefügt, dass das Element, welches zuvor das oberste war, als unterstes eingefügt wird.

Informationen zu XML-Schema-Annotationen
XML-Schema-Annotationen werden im `annotation`-Element gespeichert. Es gibt zwei Arten von Annotationen. Beide sind Elemente des annotation-Elements:

- Kompositor- oder Komponenten-Dokumentation: Enthält Informationen, die für Schema-Editoren nützlich sein könnten. Diese Informationen befinden sich im `documentation` Child-Element von `annotation`.
- Applikationsinformationen: Hier können ein Skript oder Daten eingefügt werden, die von einer verarbeitenden Applikation gelesen werden können; diese Informationen befinden

 © 2007 Altova GmbH

sich im `appinfo`-Child-Element von `annotation`..

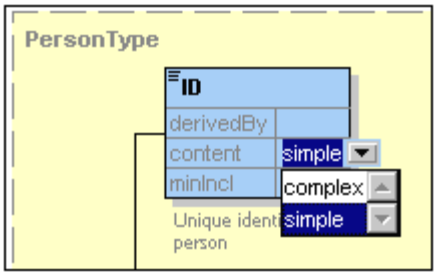
Unterhalb finden Sie den Text eines annotation-Elements. Er beruht auf dem Beispiel in der obigen Beschreibung zum Erstellen von Dokumentations- und Applikatonsinformationen.

```
<xs:element name="session_date" type="xs:dateTime" nillable="true">
   <xs:annotation>
      <xs:documentation>Datum und Zeit des
Interviews</xs:documentation>
      <xs:appinfo
source="http://www.altova.com/datehandlers/interviews">separator =
:</xs:appinfo>
   </xs:annotation>
</xs:element>
```

Konfigurieren der Content Model-Ansicht
Sie können die Content Model-Ansicht für das gesamte Schema im Dialogfeld "Schema-Ansicht konfigurieren" (**Schema-Design | Ansicht konfigurieren**) konfigurieren. Nähere Informationen zu Konfigurationsoptionen finden Sie in der Benutzerreferenz im Abschnitt Ansicht konfigurieren . Beachten Sie, dass die hier definierten Einstellungen für das gesamte Schema, die Erzeugung der Schema-Dokumentation und die Druckausgabe gelten.

Ändern von Komponenteneigenschaften direkt im Content Model
Wenn die Content Model-Ansicht so konfiguriert ist, dass Komponenten mit zusätzlichen beschreibenden Eigenschaftszeilen im Komponentenkästchen angezeigt werden, können Sie diese Informationen bearbeiten und dadurch die Eigenschaften der Komponente ändern. Sie können diese Eigenschaftszeilen durch Klicken auf das Symbolleistensymbol "Vordefinierte Details hinzufügen" ein- und ausblenden. Sie können also zwischen einer Ansicht, die die definierten Eigenschaften enthält und einer Ansicht, die diese nicht enthält hin- und herschalten. So bearbeiten Sie Komponenteneigenschaften:

- Doppelklicken Sie auf das gewünschte Informationsfeld der jeweiligen Komponente und beginnen Sie mit der Eingabe oder Bearbeitung der Daten. Wenn eine vordefinierte Option zur Auswahl steht, erscheint eine Dropdown-Liste, aus der Sie den gewünschten Eintrag auswählen können. Geben Sie andernfalls den gewünschten Wert einfach ein.



- Bestätigen Sie durch Drücken der **Eingabetaste**. Die Details-Eingabehilfe wird aktualisiert, um Ihre Änderungen anzuzeigen.

Alternativ dazu können Sie die Eigenschaften einer Komponente in der Details-Eingabehilfe ändern. Die Änderungen werden in den Platzhalterfeldern übernommen - falls diese Anzeige aktiviert ist.

Dokumentieren des Content Models
Sie können detaillierte Dokumentation zu Ihrem Schema im HTML- und MS Word-Format

erstellen. Für jede globale Komponente wird detaillierte Dokumentation generiert. Die Liste der globalen Komponenten wird im Inhaltsverzeichnis angezeigt, über das Sie Verknüpfungen zu den Content Models der einzelnen Komponenten herstellen können. Außerdem werden verwandte Elemente (wie z.B. Child-Elemente, complexTypes usw.) durch Hyperlinks referenziert, sodass Sie von einem Element zum anderen navigieren können. Um Schema-Dokumentation zu generieren, wählen Sie den Menübefehl **Schema-Design | Schema-Dokumentation generieren**.

Attribute und Identity Constraints
Attribute und Identity Constraints können in einem Fenster unterhalb der Content Model-Ansicht oder in Kästchen im Content Model selbst angezeigt werden. Durch Klicken auf die Symbole können Sie zwischen diesen beiden Ansichten wechseln. Eine Beschreibung zum Einfügen und Bearbeiten von Attributen und Identity Constraints finden Sie unter Definieren von Attributen für eine Komponente und Definieren von Identity Constraints für eine Komponente.

5.1.3 Smart Restrictions

Bei Einschränkung eines complexType mittels Restriction werden Teile des Content Models des Basistyps im abgeleiteten Typ neu geschrieben. Dies kann verwirrend sein, wenn es sich um ein komplexes Content Model handelt, da es beim Editieren des abgeleiteten Typs oft nicht einfach ist, sich genau zu erinnern, wie das Content Model des Basistyps aussieht.

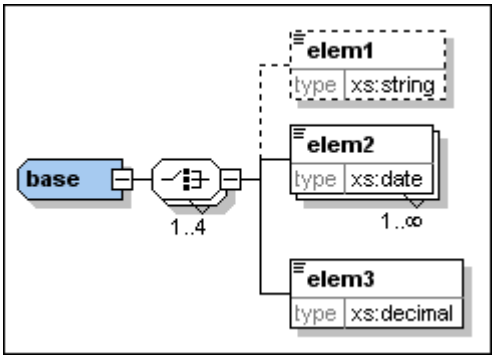
Mit Hilfe von Smart Restrictions werden die beiden Content Models in der grafischen Ansicht des abgeleiteten Content Model kombiniert und in Zusammenhang gesetzt. Im abgeleiteten complexType sehen Sie alle Partikel des Basis-complexType und in welcher Beziehung sie zum abgeleiteten Typ stehen. Außerdem bieten Smart Restrictions visuelle Anhaltspunkte und zeigen alle Möglichkeiten, wie der Basistyp mittels Restriction eingeschränkt werden kann, wodurch es einfacher wird, den abgeleiteten Typ korrekt einzuschränken.

So aktivieren Sie die Option "Smart Restrictions":

- Klicken Sie auf das Symbol "Smart Restrictions" .

Im folgenden Beispiel sehen Sie, wie sich Smart Restrictions auswirken.

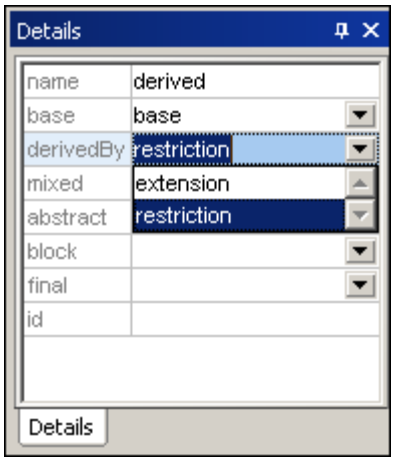
Der folgende complexType ist der Basis-Typ, der in diesem Beispiel verwendet wird:



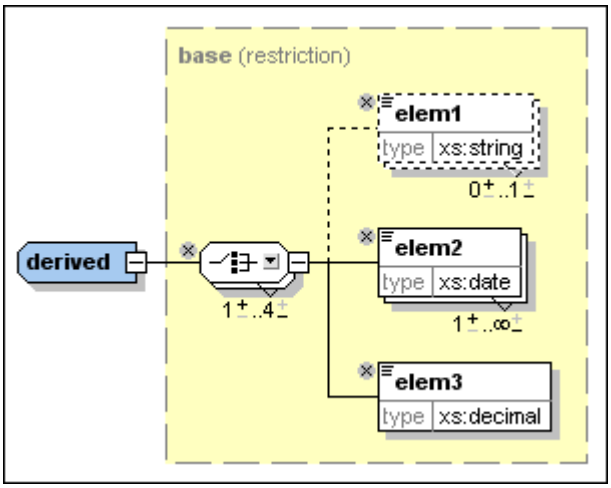
Der complexType "derived" wird folgendermaßen vom "Base" Type abgeleitet:

1. Erstellen Sie im Schema einen neuen complexType und nennen Sie ihn "derived".
2. Wählen Sie in der Eingabehilfe "Details" den Eintrag "base" aus der Dropdown-Liste **base** aus und den Eintrag "restriction" aus der Dropdown-Liste **derivedBy**.

© 2007 Altova GmbH

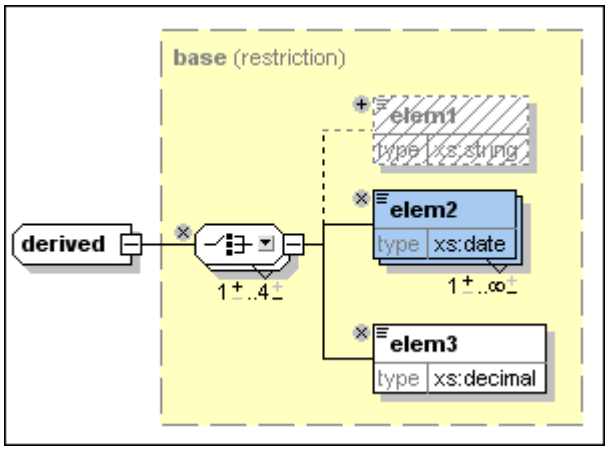


Wenn die Option "Smart Restrictions" aktiviert ist, so sieht der neue abgeleitete Typ folgendermaßen aus:

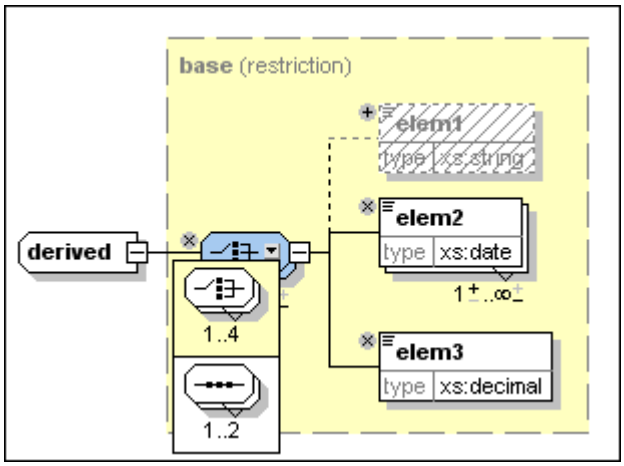


Beachten Sie die folgenden Steuerelemente, mit denen Sie den abgeleiteten Typ in diesem Beispiel durch Restriction einschränken können:

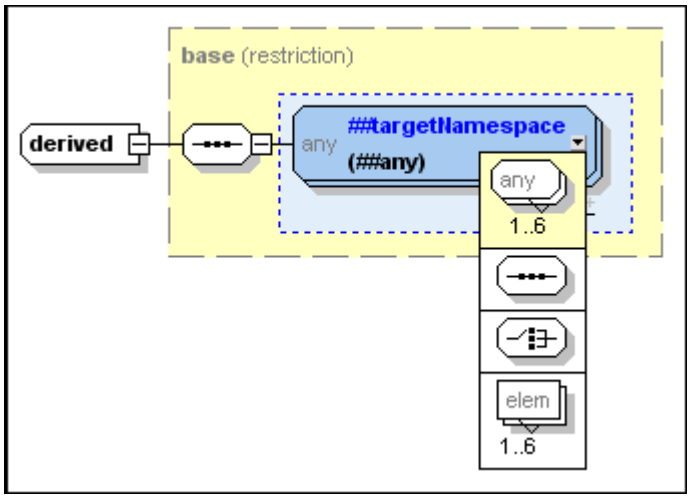
- Mit Hilfe des Symbols können Sie Elemente aus dem abgeleiteten Typ entfernen, die sich im Basis-Typ befinden. Hier wurde elem1 gelöscht. Um es erneut hinzuzufügen, klicken Sie auf dieses Symbol .



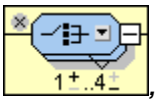
- Klicken Sie auf den Abwärtspfeil neben dem Element "Choice", um die folgende Liste aufzurufen, über die Sie die Model Group "Choice" in eine Model Group "Sequence" ändern können:

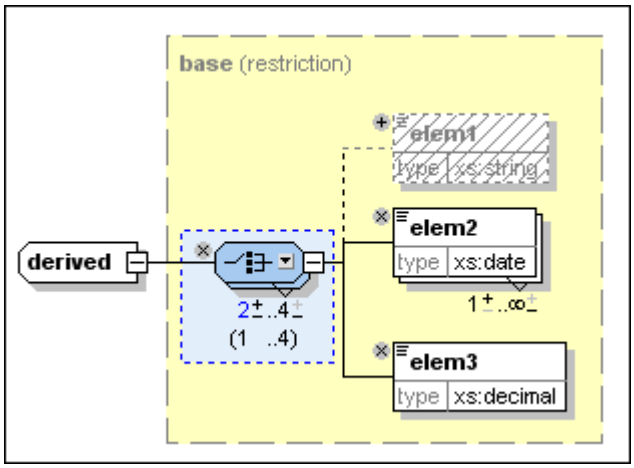


Sie können auch Wildcards auf dieselbe Art ändern - siehe Beispiel:



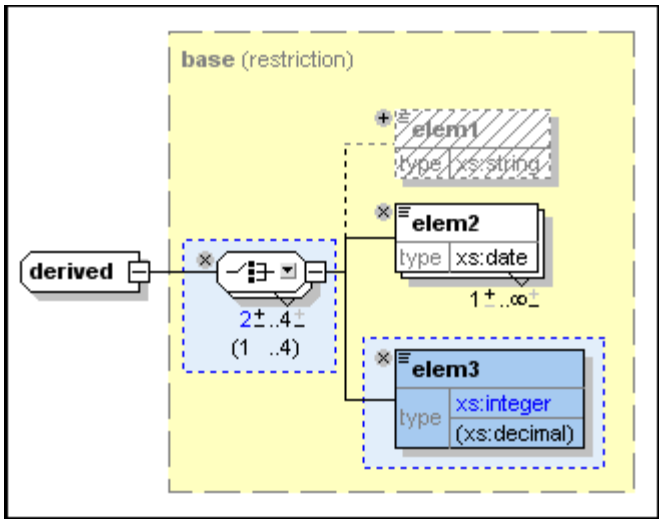
Ein ausführliche Liste darüber, welche Partikel durch welche anderen Partikel ersetzt werden können, finden Sie in der XML-Schema-Spezifikation.

- Klicken Sie auf das folgende Symbol  , um die Anzahl der "Occurrences" der Model Group zu ändern. Durch Klicken auf das +-Zeichen über der 1 können Sie die Mindestanzahl der Occurrences erhöhen, durch Klicken auf das Minus-Zeichen unterhalb von "4" können Sie die Maximalanzahl der Occurrences verringern. Diese Steuerelemente werden angezeigt, wenn es sich beim Occurrence-Bereich um einen realen Bereich (z.B. 2-5) handelt und nicht um einen bestimmten Wert (z.B. 4-4). Sie werden auch angezeigt, wenn der Occurrence-Bereich falsch ist.

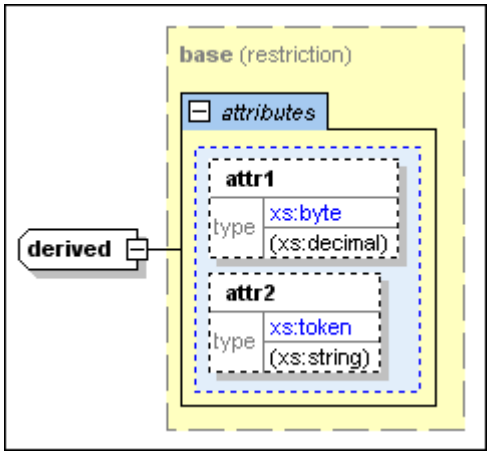


Hier sehen Sie, dass die Minimum Occurrence für dieses Element in 2 geändert wurde. Beachten Sie, dass die Model Group nun einen blauen Hintergrund aufweist, was bedeutet, dass sie nicht mehr mit der Model Group im Base complexType identisch ist. Außerdem wird der Occurrence-Bereich der Model Group im Basiselement nun in Klammern angezeigt.

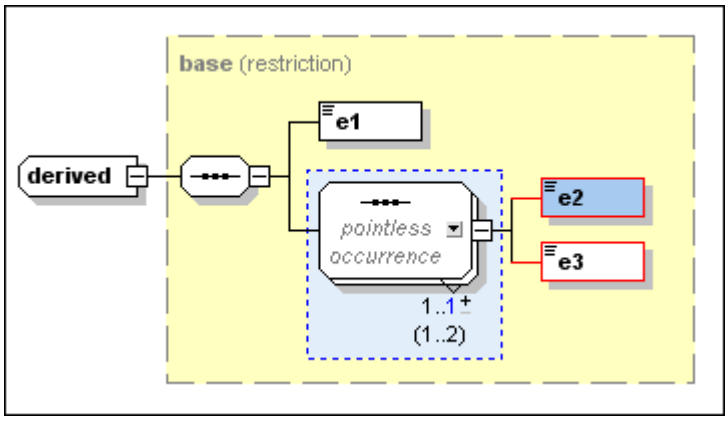
- Sie können die Datentypen von Attributen oder Elementen ändern, wenn es sich beim neuen Datentyp um eine gültige Restriction des in der XML Schema-Spezifikation definierten Basis-Typs handelt. So können Sie z.B. den Datentyp von elem3 im Datentyp "derived" von Decimal in Integer ändern. Das Element wird danach mit blauem Hintergrund angezeigt, um hervorzuheben, dass es nicht mit dem Element im Basis-Typ identisch ist. Der Typ, den das Element im Basis-Typ hat, wird in Klammern angezeigt:



In diesem Beispiel sehen Sie Attribute, deren Datentypen im abgeleiteten complexType durch Restriction eingeschränkt wurden.



- Smart Restrictions informieren Sie auch über *überflüssige Vorkommen* (pointless occurrences) im Content Model. Ein "überflüssiges Vorkommen" wird z.B. dann angezeigt, wenn eine im Content Model vorhandene Sequenz unnötig ist. In diesem Beispiel sehen Sie ein solches "überflüssiges Vorkommen":



Bitte beachten Sie: Überflüssige Vorkommen werden nur angezeigt, wenn das Content Model einen Fehler enthält. In manchen Fällen enthält ein Content Model ein "überflüssiges Vorkommen" ist aber trotzdem gültig. In diesen Fällen wird das überflüssige Vorkommen nicht explizit angezeigt, um nicht zu Verwirrung zu führen.

Nähere Informationen über "überflüssige Vorkommen" (Pointless occurrences) finden Sie in der XML-Schema-Spezifikation.

5.1.4 Arbeiten mit SchemaAgent

XMLSpy kann für die Verwendung mit Altova SchemaAgent konfiguriert werden.

SchemaAgent Technologie
SchemaAgent gestattet Benutzern, Beziehungen zwischen mehreren Schemas zu erstellen und zu bearbeiten. SchemaAgent besteht aus:

© 2007 Altova GmbH

- Einem SchemaAgent Server, der Informationen über die Beziehungen zwischen Schemas in einem oder mehreren von Ihnen definierten Suchpfaden (Ordnern auf dem Netzwerk) enthält.
- Einem SchemaAgent Client, dem Altova Produkt SchemaAgent 2007, der anhand der Schema-Informationen vom SchemaAgent, mit dem er verbunden ist, (i) Beziehungen zwischen diesen Schemas erstellt und (ii) diese Schemas verwaltet (durch Umbenennen, Verschieben, Löschen von Schemas usw.);

Es stehen zwei Arten von SchemaAgent Servern zur Verfügung:

- Altova SchemaAgent Server 2007, der auf einem Netzwerk installiert werden kann, und auf den Sie über das Netzwerk zugreifen können und
- Altova SchemaAgent 2007. Dies ist das SchemaAgent Client-Produkt. Es enthält eine Leichtversion des Servers, den so gennanten LocalServer, der nur auf demselben Rechner verwendet werden kann, auf dem auch SchemaAgent installiert ist.

XMLSpy verwendet die SchemaAgent-Technologie zum direkten Editieren von Schemas in der Schema-Ansicht, wobei das Programm Informationen über andere Schemas verwendet, die es von einem SchemaAgent Server erhält. In dieser Konfiguration ist XMLSpy mit einem SchemaAgent Server verbunden und sendet über SchemaAgent Client Requests an SchemaAgent Server. Wenn XMLSpy für die Arbeit mit SchemaAgent konfiguriert wurde, enthalten die Eingabehilfen in der Schema-Ansicht nicht nur Komponenten aus dem derzeit in der Schema-Ansicht gerade aktiven Schema sondern auch Komponenten aus anderen Schemas in den Suchpfaden des SchemaAgent Servers, mit dem das Programm gerade verbunden ist. Auf diese Art haben Sie direkten Zugriff auf diese Komponenten. Sie können das Content Model einer zu einem anderen Schema gehörenden Komponente in der Schema-Ansicht anzeigen und sie mit oder ohne Änderungen wiederverwenden. Außerdem können Sie Beziehungen zwischen Schemas erstellen und dadurch komplexe Schemas direkt in XMLSpy in Module aufteilen und verwalten.

Installation von SchemaAgent und SchemaAgent Server

Nähere Informationen zu SchemaAgent und SchemaAgent Server und zum Konfigurieren von Suchpfaden auf Servern finden Sie im Benutzerhandbuch zu SchemaAgent.

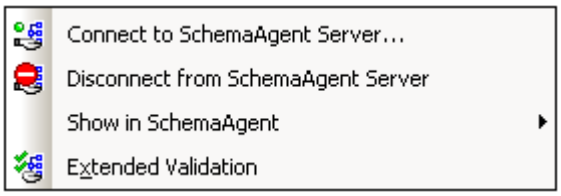
Einrichten von XMLSpy als SchemaAgent Client
Um XMLSpy für die Arbeit als SchemaAgent Client zu konfigurieren, gehen Sie folgendermaßen vor:

- Downloaden Sie SchemaAgent von der Altova Website und installieren Sie ihn auf Ihrem Netzwerk. Sie können nun über den LocalServer von SchemaAgent auf Schemas zugreifen. Informationen zum Konfigurieren von Suchpfaden in LocalServer finden Sie im SchemaAgent-Benutzerhandbuch. **Beachten Sie**, dass Sie für SchemaAgent eine Lizenz benötigen, die Sie nach Ablauf des Testzeitraums erwerben müssen, um das Produkt weiterverwenden zu können. Beachten Sie, dass die Altova Altova MissionKit-Pakete das Produkt SchemaAgent sowie einen Lizenzschlüssel dazu enthalten. (Die SchemaAgent Server-Applikation ist jedoch nicht in den Altova MissionKit-Paketen enthalten.)
- Zusätzlich sollten Sie bei Bedarf den netzwerkbasierten SchemaAgent Server von der Altova Website herunterladen und installieren.
- Definieren Sie den/die Suchpfad(e) für den SchemaAgent Server (Konfigurieren von SchemaAgent Server). Eine ausführliche Erläuterung dazu finden Sie in der Dokumentation zu SchemaAgent. (Ein Suchpfad ist ein Pfad zu dem Ordner, der die XML-Schemas enthält, die miteinander in Zusammenhang stehen und gemappt werden).
- Stellen Sie eine Verbindung zwischen XMLSpy und dem SchemaAgent server.

Wichtig: Alle SchemaAgent-Produkte und alle mit SchemaAgent in Zusammenhang stehenden Produkte von Altova (einschließlich XMLSpy) beginnend mit Version 2005 Release 3 sind **nicht** mit früheren Versionen von SchemaAgent oder mit SchemaAgent in Zusammenhang stehenden Produkten kompatibel.

SchemaAgent-Befehle in XMLSpy

Die SchemaAgent-Funktion in XMLSpy steht nur in der Schema-Ansicht zur Verfügung und wird über die Menübefehle im Schema-Design-Menü *(siehe Abbildung unten),* die Symbolleiste und die Eingabehilfen in der Schema-Ansicht aufgerufen.



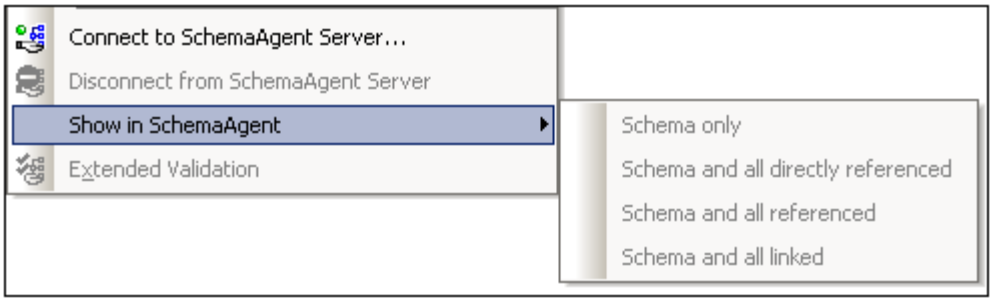
Über die Menübefehle stehen Ihnen allgemeine Verwaltungsfunktionen zur Verfügung. Die Eingabehilfen (und die Standardfunktion der Benutzeroberfläche wie z.B. Drag & Drop) dienen zum Bearbeiten von Schemas.

In diesem Abschnitt wird beschrieben, wie man die in der Schema-Ansicht zur Verfügung stehenden SchemaAgent-Funktionen verwendet.

Herstellen einer Verbindung mit SchemaAgent Server

Bitte beachten Sie: Um eine Verbindung herstellen zu können, muss SchemaAgent Client installiert sein.

Bevor Sie eine Verbindung zu SchemaAgent Server hergestellt haben, ist im Menü **Schema-Design** nur der Befehl **Verbindung zu SchemaAgent Server herstellen,** aktiv. Alle anderen SchemaAgent-Befehle im Menü **Schema-Design** sind deaktiviert (*siehe Abbildung unten*). Die anderen Menüeinträge werden aktiv, sobald eine Verbindung zu SchemaAgent Server besteht.



Bitte beachten Sie: Wenn SchemaAgent Client nicht installiert ist, kann keine Verbindung zu SchemaAgent Server hergestellt werden.

Schritte beim Herstellen einer Verbindung

So stellen Sie eine Verbindung zu einem SchemaAgent Server her:

1. Klicken Sie auf des Symbol "Verbindung zu SchemaAgent Server herstellen" **(Schema-Design | Verbindung zu SchemaAgent Server herstellen).** Daraufhin wird das gleichnamige Dialogfeld angezeigt (*siehe Screenshot unten*).

 © 2007 Altova GmbH

2. Sie können entweder mit dem lokalen Server arbeiten (dem SchemaAgent, Server, der mit Altova SchemaAgent 2007 mitgeliefert wird). oder mit einem Netzwerk-Server (dem Altova SchemaAgent Server, der kostenlos erhältlich ist). Wenn Sie die Option "Lokal arbeiten" auswählen, wird der lokale Server von SchemaAgent gestartet, wenn Sie auf OK klicken und es wird eine Verbindung mit diesem Server hergestellt. Wenn Sie die Option "Verbindung zu Netzwerk-Server herstellen" wählen, muss der ausgewählte SchemaAgent Server ausgeführt werden, damit eine Verbindung hergestellt werden kann.

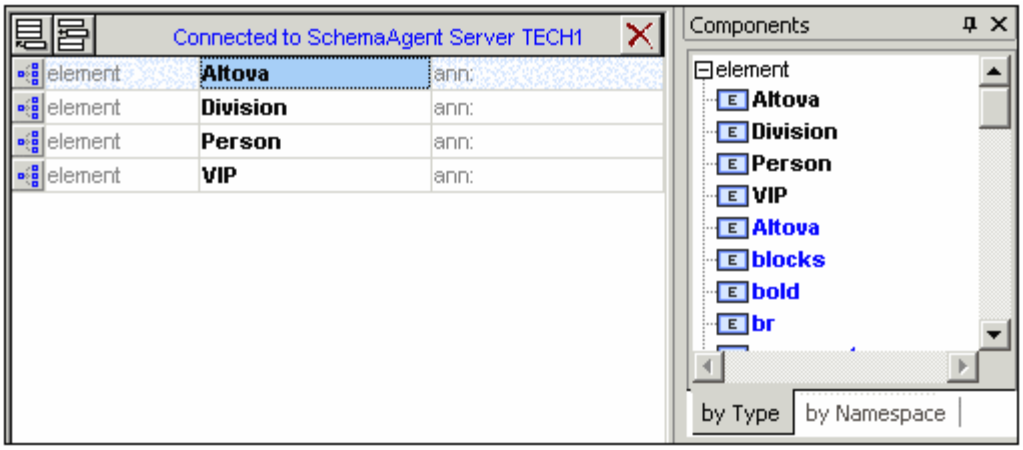
Bitte beachten Sie zu Servern, die unter Windows XP SP2 laufen
Wenn der Name des SchemaAgent Servers im Dialogfeld "Verbindung zu SchemaAgent Server herstellen" aufgelistet ist, Sie jedoch keine Verbindung herstellen können, liegt dies möglicherweise daran, dass Ihr Server nicht an der Namensauflösung in Ihrem Netzwerk teilnimmt. Die Namensauflösung wird durch die Standardeinstellungen der Windows XP SP2 Firewall verhindert.

Um die Verbindung zu einem derartigen Server herzustellen, haben Sie folgende Möglichkeiten:

- Ändern Sie die Servereinstellungen, um die Namensauflösung zuzulassen oder
- Geben Sie im Bearbeitungsfeld des Dialogfelds zur Verbindungsherstellung die IP-Adresse des Servers ein.

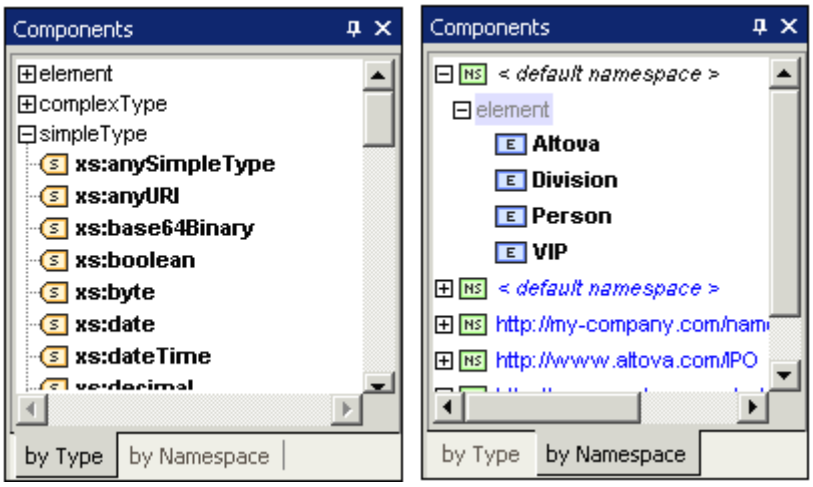
Sie müssen diese Einstellungen nur einmal vornehmen, da SchemaAgent Client den Connection String der letzten funktionierenden Verbindung speichert.

Schema/WSDL-Ansicht nach Herstellen der Verbindung mit SchemaAgent Server
Nachdem Sie eine Verbindung zu einem SchemaAgent Server hergestellt haben, sieht die Schema/WSDL-Ansicht in etwa folgendermaßen aus:



Beachten Sie bitte Folgendes:

- In der Schema/WSDL-Ansicht werden nun in der Ansicht der globalen Elemente der Text "Verbunden mit SchemaAgent Server ..." mit Angabe des Servers, mit dem Sie eine Verbindung hergestellt haben.
- Sie haben nun Zugriff auf alle Schemas und Schema-Konstrukte, die unter dem Server-Suchpfad zur Verfügung stehen. SchemaAgent Schema-Konstrukte: Globale Elemente, complexTypes, simpleTypes usw. werden unterhalb der Konstrukte des aktiven Schemas **(schwarz, fett) blau und fett** angezeigt.

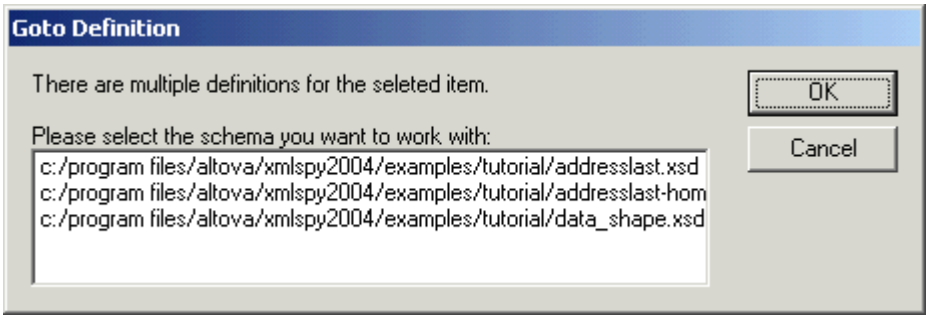


Schema-Konstrukte können in der Schema/WSDL-Ansicht nach Typ oder nach Namespace (Register "by Type" und "by Namespace") angezeigt werden.

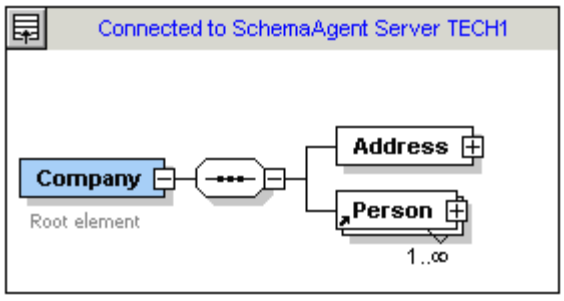
Öffnen von Schemas aus dem Suchpfad

In diesem Beispiel wird gezeigt, wie man ein Schema öffnet, das unter einem in SchemaAgent Server definierten Suchpfad gespeichert ist. Als aktives Schema wird die Datei `DB2schema.xsd` aus dem Ordner `..\Tutorial` verwendet. Das Register `By Type` der Komponenten-Eingabehilfe ist das aktive Register.

1. Scrollen Sie in der Komponenten-Eingabehilfe hinunter zum blauen Eintrag `Company` in der Komponente "Element" und doppelklicken Sie darauf. Daraufhin wird das Dialogfeld "Gehe zu Definition" geöffnet.



2. Klicken Sie auf den Eintrag `Addresslast.xsd` und bestätigen Sie mit OK. Das Schema `addresslast.xsd` wird geöffnet und das Content Model des Elements `Company` wird angezeigt.



 © 2007 Altova GmbH

Bitte beachten Sie: Wenn Sie auf ein SchemaAgent Schema-Konstrukt, wie z.B. ein Element, einen complexType oder einen simpleType doppelklicken, wird das dazugehörige Schema (sowie alle anderen inkludierten Schemas) in XMLSpy geöffnet.

Verwenden von Schema-Konstrukten

XML-Schema bietet drei grundlegende Methoden, um Schemas in Module aufzugliedern: Import, Include und Redefine (IIR). Bei jeder Methode gelten andere Namespace-Anforderungen. Diese werden jedoch automatisch beim Erstellen von IIRs von SchemaAgent Client und XMLSpy überprüft.

Import-, Include- und Redefine-Anweisungen (IIR)

Es gibt unterschiedliche Arten, um ein Schema-Konstrukt "einzufügen":

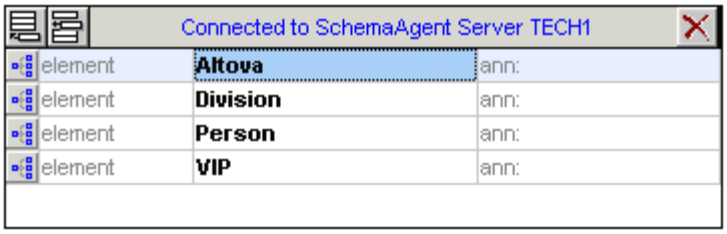
- Sie können globale Elemente direkt aus der Komponenten-Eingabehilfe (in der Schema/WSDL Design-Ansicht) in das Content Model einer Schemakomponente ziehen
- Komponenten, wie z.B. complexTypes und simpleTypes können aus einem Listenfeld ausgewählt werden, das automatisch geöffnet wird, wenn Sie ein neues Element/Attribut usw. definieren.
- Komponenten wie z.B. complexTypes können bei der Erstellung/Aktualisierung dieser Konstruktart in der Details-Eingabehilfe ausgewählt werden.

Integrieren von Schema-Komponenten

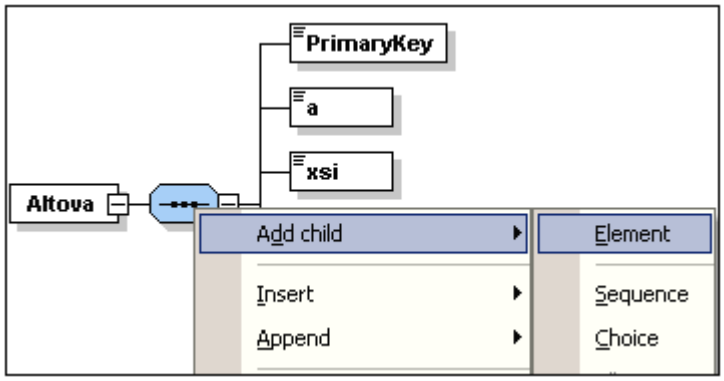
In diesem Beispiel wird die Datei `DB2schema.xsd` aus dem Ordner `..\Tutorial` als aktives Schema verwendet; das Register `By Type` der Komponenten-Eingabehilfe ist aktiv.

So verwenden Sie Schema-Konstrukte aus SchemaAgent Server Schemas:

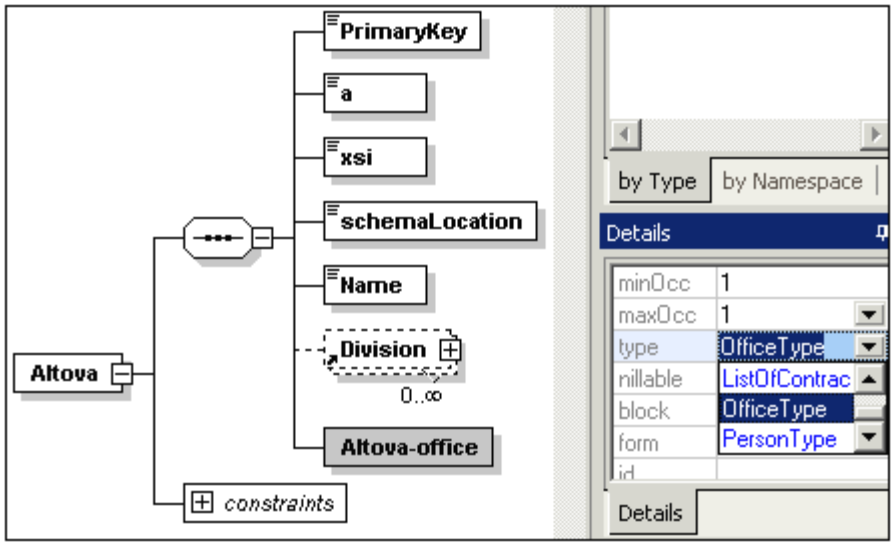
1. Stellen Sie sicher, dass eine Verbindung mit einem SchemaAgent Server hergestellt ist (siehe Herstellen einer Verbindung mit SchemaAgent Server).
2. Öffnen Sie die Datei `DB2Schema.xsd` und benennen Sie sie für dieses Beispiel in `Altova-office` um.



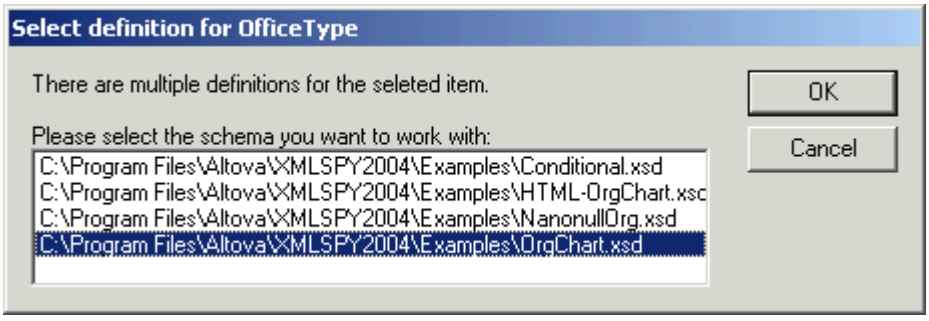
3. Klicken Sie in der Schema-Übersicht auf das Element `Altova`, um dessen Content Model aufzurufen.
4. Rechtsklicken Sie auf den `Altova` sequence-Kompositor und wählen Sie die Menüoption **Child hinzufügen | Element**. Beachten Sie, dass nun automatisch ein Listenfeld mit allen globalen Elementen aus dem Serverpfad geöffnet wird. Wenn Sie eines auswählen, wird dieses Element integriert.



5. Geben Sie `Altova-office` als Namen für dieses neue Element ein und drücken Sie die Eingabetaste.
6. Klicken Sie in der Details-Eingabehilfe auf das Listenfeld `type` und wählen Sie den Eintrag `OfficeType` aus.

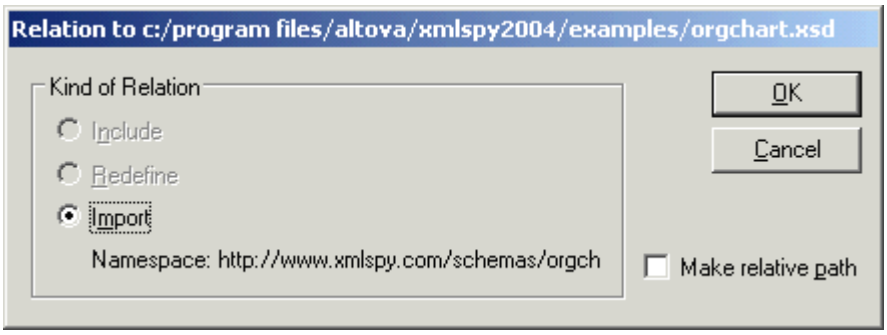


Daraufhin wird das Dialogfeld "Definition für OfficeType auswählen" geöffnet.

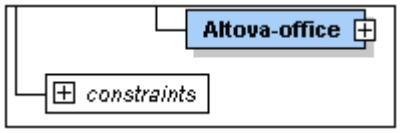


7. Wählen Sie `Orgchart.xsd` und klicken Sie auf OK.

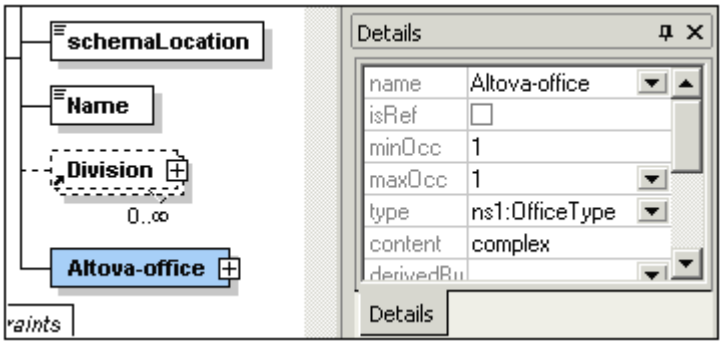
 © 2007 Altova GmbH



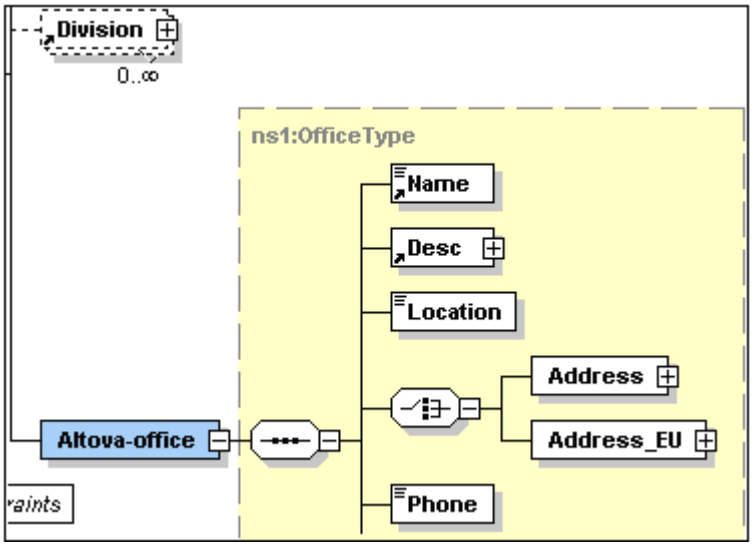
8. Klicken Sie im nächsten Dialogfeld auf OK. Der Befehl "Import" wurde automatisch ausgewählt. Das Element `Altova-office` weist nun ein Plus-Zeichen zum Erweitern auf.



Bitte beachten Sie, dass sich der Eintrag `type` entry in der Details-Eingabehilfe geändert hat; er wird nun als `ns1:OfficeType` angezeigt, da die Schemadatei `Orgchart.xsd` importiert wurde und die beiden Schemas unterschiedliche Target Namespaces haben müssen. Außerdem wurde zum Schema ein Import-Befehl hinzugefügt.



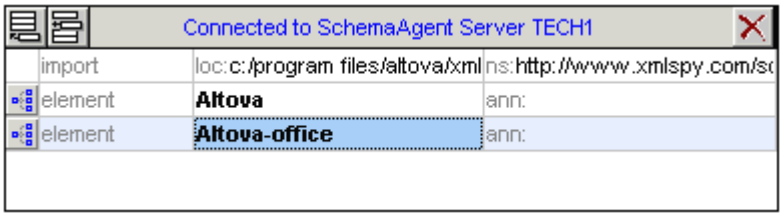
9. Klicken Sie auf das Erweiterungssymbol, um das Content Model von `OfficeType` anzuzeigen.



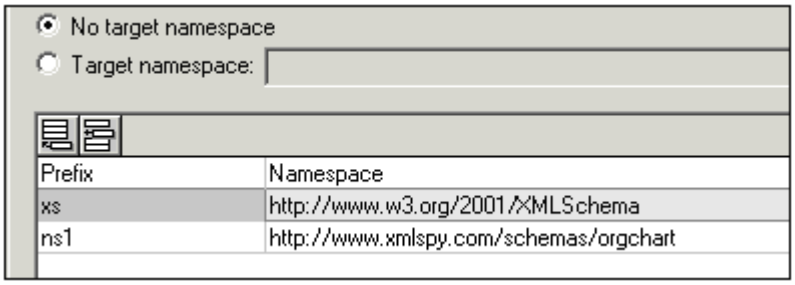
10. Drücken Sie F8, um das Schema zu validieren. Daraufhin sollte die Meldung, dass das Schema gültig ist, erscheinen.

Aufräumen des Schemas:

1. Löschen Sie das Element `Division` im Content Model.
2. Klicken Sie auf das Symbol zum Anzeigen der globalen Elemente, um in die Schema-Übersicht zu wechseln.
3. Löschen Sie die folgenden globalen Elemente: `Division`, `Person` und `VIP`.

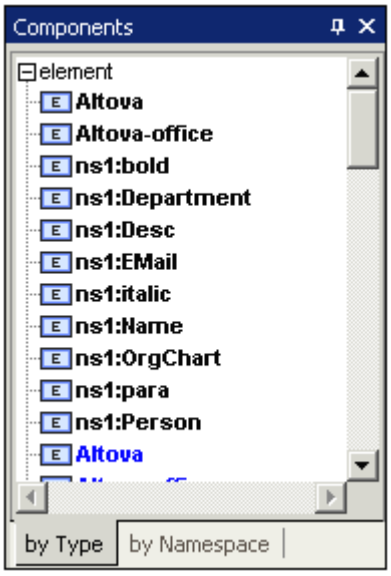


4. Wählen Sie die Menüoption **Schema-Design | Schema-Einstellungen**, um zu sehen, inwiefern sich die Namespace-Einstellungen geändert haben.



Das `ns1`-Präfix wurde automatisch zum Namespace `www.xmlspy.com/schemas/orgchart` hinzugefügt. Die Komponente- und die Details-Eingabehilfe zeigen alle importierten Konstrukte mit dem ns1: namespace Präfix an.

 © 2007 Altova GmbH

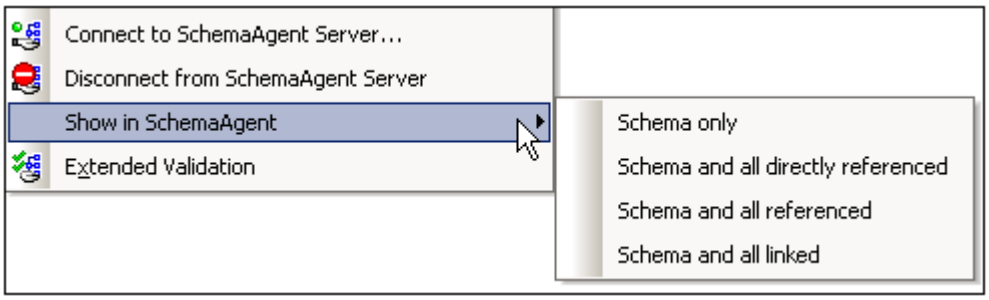


Bitte beachten Sie:

- Änderungen, die in XMLSpy unter Verwendung von SchemaAgent vorgenommen wurden, werden automatisch in allen anderen Schemas im SchemaAgent Serverpfad, der das geänderte Schema referenziert, übernommen.
- Unter Umständen werden Konstrukte wie Elemente, simpleTypes usw. in den Eingabenhilfen doppelt angezeigt (in Schwarz und Blau), wenn sich das Schema, mit dem Sie gerade arbeiten, auch unter dem SchemaAgent Serverpfad befindet.

Anzeigen von Schemas in SchemaAgent

Um mit dem aktiven Schema und den damit verknüpften Schemas in SchemaAgent zu arbeiten, klicken Sie auf die Menüoption **Schema-Design | In SchemaAgent anzeigen | *Schema und alle damit verbundene*** *(siehe Abbildung unten).*



Sie haben die Option, entweder nur das aktive Schema in SchemaAgent zu öffnen oder das aktive Schema zusammen mit entweder (i) allen direkt referenzierten Schemas oder (ii) allen referenzierten Schemas oder (iii) allen verbundenen Schemas zu öffnen.

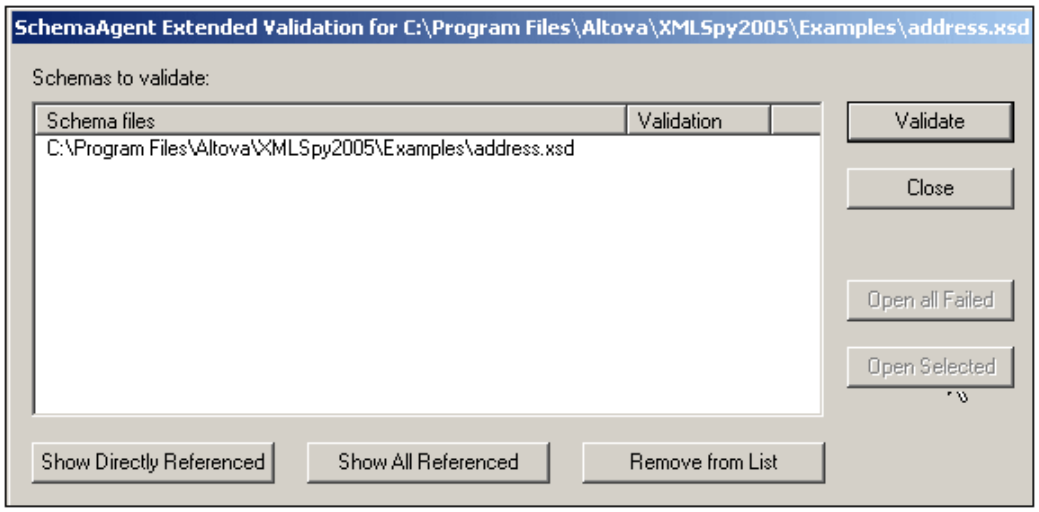
Erweiterte Validierung

XMLSpy gestattet Ihnen in Verbindung mit SchemaAgent nicht nur das aktive Schema zu validieren, sondern auch Schemas, die mit dem aktiven Schema in Zusammenhang stehen.

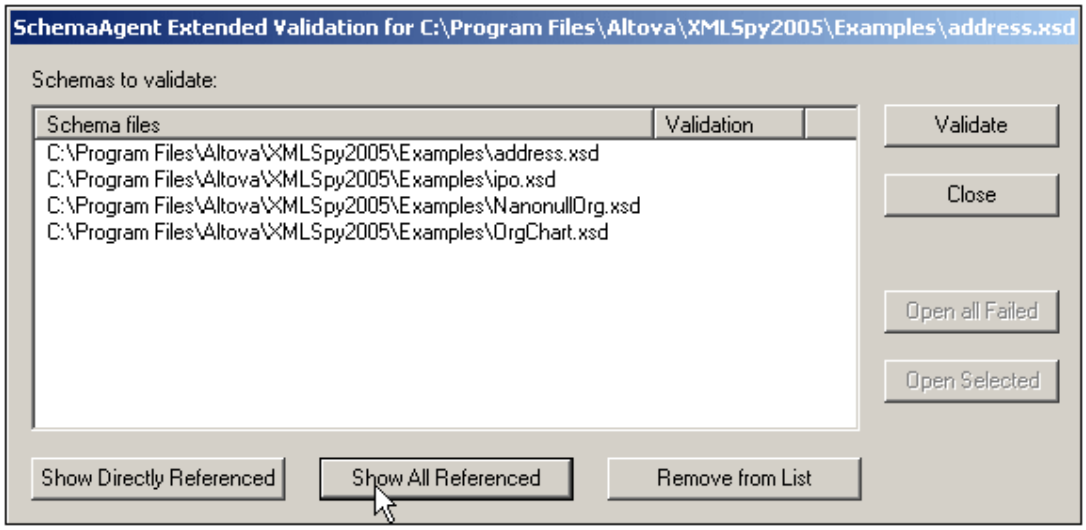
SchemaAgent unterscheidet bei der erweiterten Validierung zwischen: (i) direkt referenzierten Schemas und (ii) referenzierten Schemas.

Die Funktionsweise der erweiterten Validierung wird im Folgenden anhand eines Beispiels beschrieben. In diesem Beispiel wird als aktive Schemadatei in der Schema/WSDL-Ansicht von XMLSpy `address.xsd` verwendet. Gehen Sie bei der erweiterten Validierung vor wie folgt:

1. Klicken Sie in der Symbolleiste auf das Symbol "Erweiterte Validierung" oder wählen Sie den Menüeintrag **Schema-Design | Erweiterte Validierung**. Daraufhin wird das Dialogfeld "Erweiterte Validierung" geöffnet. Hier können Sie angeben, ob nur das aktive Schema oder auch ein oder mehrere damit verknüpfte Schemas validiert werden sollen.



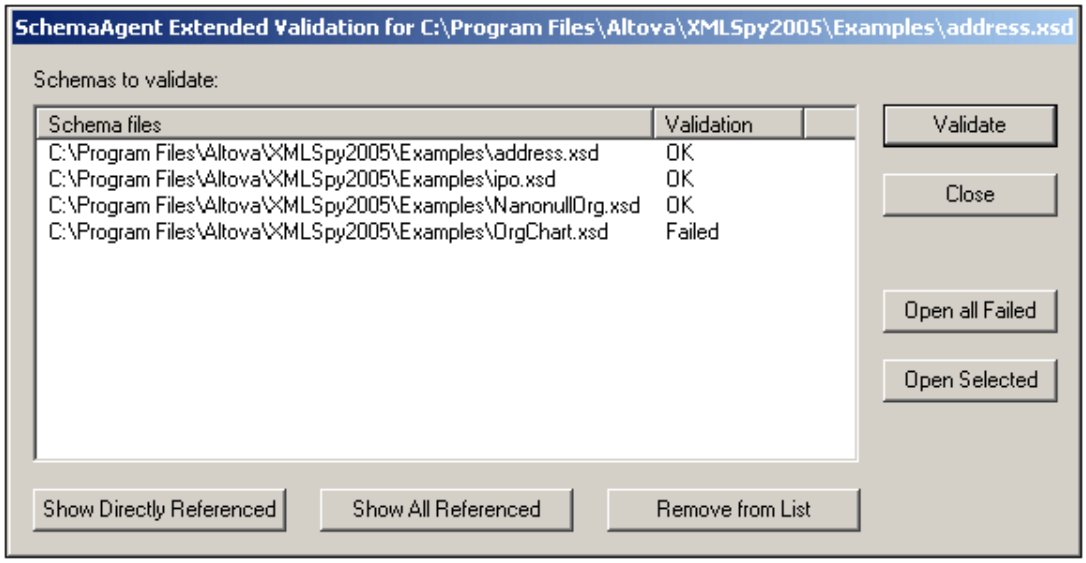
2. Um Schemas zur Liste hinzuzufügen, klicken Sie je nach Bedarf entweder auf "Direkt referenzierte anzeigen" oder auf "Alle referenzierten anzeigen". In diesem Beispiel haben wir die Option "Alle referenzierten anzeigen" gewählt, woraufhin alle referenzierten Dateien in die Liste übernommen werden.



Sie können zu diesem Zeitpunkt ein Schema aus der Liste entfernen ("Aus Liste löschen'").

3. Klicken Sie auf die Schaltfläche "Validieren", um alle Schemas im Listenfeld zu

validieren.

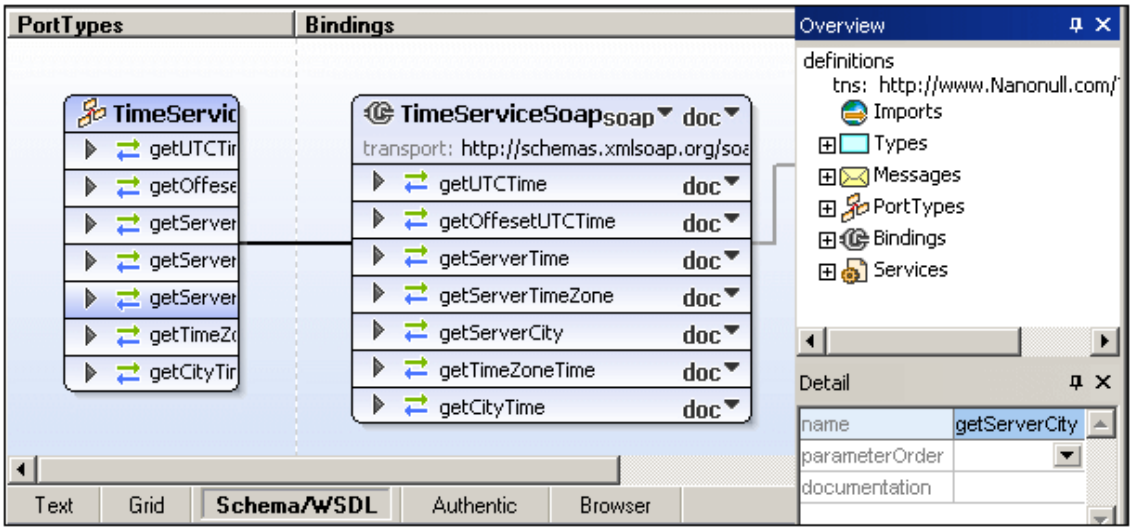


In der Spalte "Validieren" sehen Sie, ob die Validierung erfolgreich war oder fehlgeschlagen ist.

Sie können nun die Schemas, bei denen ein Validierungsfehler aufgetreten ist oder eine Gruppe von ausgewählten Schemas in XMLSpy öffnen.

5.2 WSDL Design-Ansicht

In der WSDL Design-Ansicht (*Abbildung unten*) sehen Sie eine Benutzeroberfläche zum grafischen Editieren von WSDL-Dokumenten. Die Struktur und Komponenten eines WSDL-Dokuments werden mit Hilfe von grafischen Mitteln im Hauptfenster erstellt. Über die Eingabehilfen stehen zusätzliche Editiermöglichkeiten zur Verfügung.



Das Hauptfenster (bestehend aus den Abschnitten PortTypes, Bindings und Services) und die Eingabehilfen (Übersicht und Detail) werden in den Unterabschnitten dieses Abschnitts näher beschrieben. (Eine Beschreibung zum Arbeiten mit Projekten finden Sie in der Benutzerreferenz unter Menü "Projekt".)

Funktionen in der WSDL-Ansicht
In der WSDL-Ansicht stehen die folgenden Funktionen zur Verfügung:

- Grafische Anzeige aller WSDL-Elemente im Hauptfenster, gruppiert nach PortTypes, Bindings und Services sowie eine Anzeige der WSDL-Dateistruktur im Hauptfenster.
- Direkte Bearbeitung von WSDL-Elementen mittels Drag & Drop.
- Die Möglichkeit, WSDL-Elemente, die in der grafischen Ansicht zu sehen sind, hinzuzufügen, anzuhängen und zu löschen (kontextsensitives Menü).
- Die Möglichkeit, Werte in der Details-Eingabehilfe einzugeben und zu bearbeiten.
- WSDL-Validierung gegen einen W3C Working Draft.
- Automatisches Wechseln in die Schema Design-Ansicht, wenn sich ein referenziertes oder importiertes Schema als ungültig herausstellt.
- Bearbeiten von Schema-Typen in der WSDL Design-Ansicht.
- Generierung von WSDL-Dokumentation in MS Word oder HTML.
- Generierung eines Diagramms (PNG-Bild) des WSDL-Dokuments im Hauptfenster.

Anzeige von Dateien
Beachten Sie die folgenden Punkte:

- Wenn Sie eine WSDL-Datei öffnen, wird diese automatisch in der WSLD-Design-Ansicht geöffnet.
- Sie können ein WSDL-Dokument auch in der Text-Ansicht und der Enhanced Grid-Ansicht öffnen. Klicken Sie dazu auf das entsprechende Register.
- Wenn die WSDL-Datei eine Referenz zu einem XML-Schema enthält, kann das Schema durch Auswahl des Befehls **WSDL | Types | Schema(s) in Schema-Ansicht**

© 2007 Altova GmbH

bearbeiten angezeigt und bearbeitet werden. Daraufhin wird die Schema-Datei in der Schema Design-Ansicht geöffnet.

- Wenn eine mit der Datei verknüpfte Schema-Datei geöffnet ist, können Sie die Ansicht der WSDL-Datei nicht wechseln (z.B. von der WSDL Design Ansicht in die Text-Ansicht). Stellen Sie daher sicher, dass Sie die Änderungen an der Schema-Datei gespeichert und die Datei geschlossen haben, bevor Sie die Ansicht für die WSDL-Datei wechseln.

Die Benutzeroberfläche im Überblick

Projekt- und Info-Fenster

- Im Projekt-Fenster sehen Sie die Dateien eines Projekts. Damit können Sie ein Projekt besser verwalten. Nähere Informationen zum Arbeiten mit Projekten finden Sie in der Benutzerreferenz unter dem Kapitel Menü "Projekt".
- Im Info-Fenster sehen Sie nähere Informationen über das im Hauptfenster markierte Element oder Attribut.

Hauptfenster

In diesem Bereich wird die WSDL-Datei bearbeitet. Beachten Sie, was die Dateiansicht betrifft, die folgenden Punkte:

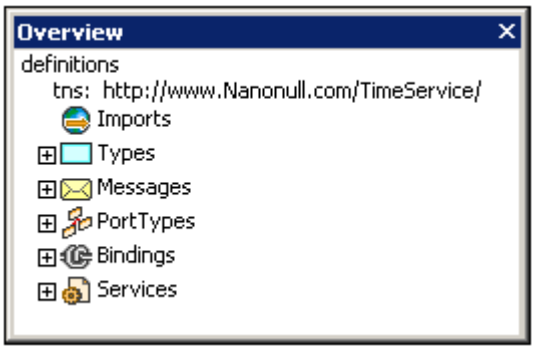
- Wenn Sie eine WSDL-Datei öffnen, wird diese automatisch in der WSDL Design-Ansicht geöffnet.
- Sie können eine WSDL-Datei auch in der Text- und der Enhanced Grid-Ansicht anzeigen. Klicken Sie dazu auf das entsprechende Register.
- Wenn die WSDL-Datei eine Referenz zu einem XML-Schema enthält, können Sie dieses Schema durch Auswahl von **WSDL | Typen | Schema bearbeiten** öffnen und bearbeiten. Die Schemadatei wird in der Schema Design-Ansicht geöffnet.
- Wenn eine verknüpfte Schemadatei geöffnet ist, darf die Ansicht der WSDL-Datei nicht gewechselt werden (z.B. von der WSDL Design-Ansicht in die Textansicht). Stellen Sie vor dem Wechseln der Ansicht für die WSDL-Datei sicher, dass Sie die Änderungen an der Schemadatei gespeichert haben und die Datei geschlossen haben.

Nähere Informationen zum Operations-, PortTypes-, Bindings-, and Services-Fenster finden Sie unter WSDL Design-Komponenten.

Eingabehilfen

Es gibt zwei Eingabehilfen: Das Übersichtsfenster und das Details-Fenster.

Übersichtsfenster



Das Übersichtsfenster enthält eine Übersicht über die Struktur des WSDL-Dokuments. Sie sehen darin die Komponenten der WSDL-Datei in Form einer Baumstruktur in der allgemeinen Reihenfolge, in der sie im Text vorkommen. Sie können die Komponenten erweitern und

reduzieren, um die Sub-Komponenten ein- bzw. auszublenden. Wenn Sie im Übersichtsfenster eine Komponente auswählen, werden ihre Eigenschaften im Details-Fenster angezeigt, wo sie bearbeitet werden können.

Details-Fenster

Im Details-Fenster werden Komponenten des WSDL-Dokuments sowie deren Eigenschaften einzeln angezeigt. Sie können diese Eigenschaften im Details-Fenster bearbeiten. Um eine Komponente (und ihre Eigenschaften) im Details-Fenster anzuzeigen, wählen Sie diese entweder im Hauptfenster oder im Übersichtsfenster aus.

5.2.1 Hauptfenster

Im Hauptfenster wird das WSDL-Dokument bearbeitet. Es ist vertikal in drei Abschnitte unterteilt: Port Types, Bindings und Services. Die Beziehung zwischen einem PortType und einem Binding und einem Binding und einem Service wird jeweils durch Konnektorlinien angezeigt. Jeder dieser drei Abschnitte wird weiter unten ausführlich beschrieben.

Symbole im Hauptfenster
Die folgenden Symbole werden im Hauptfenster verwendet:

PortType

Binding

Service

Operation

Message

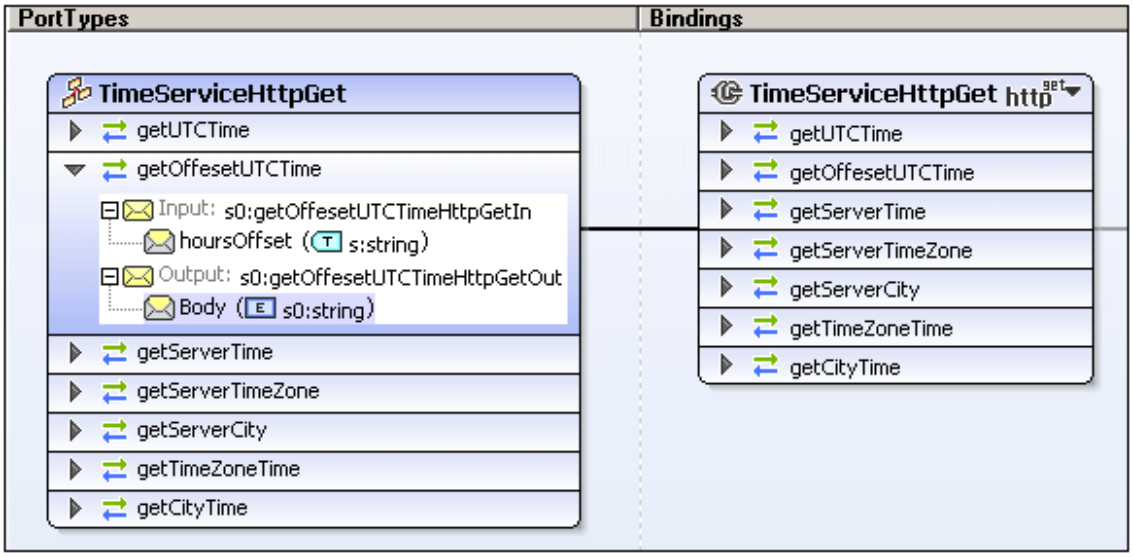
Message Part (Parameter)

XSD-Element

XSD simpleType oder complexType

Port

PortTypes
Der Abschnitt "PortTypes" enthält alle PortTypes, die im WSDL-Dokument definiert sind (*in der Abbildung unten sehen Sie nur einen portType im Abschnitt PortTypes*).

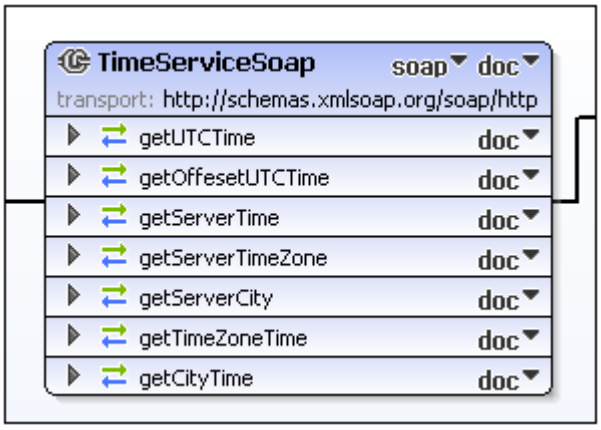


Im Fenster "PortTypes" werden alle PortTypes der WSDL-Datei angezeigt. Jeder PortType wird grafisch in Form eines Kästchens dargestellt, das die Operationen (Operations) für diesen PortType enthält. Operationen können erweitert angezeigt werden, sodass Sie deren Messages sehen. Kicken Sie dazu auf das Symbol links vom Operationsnamen. Wenn eine Message einen Message Part enthält, kann sie erweitert werden, sodass der Message Part zu sehen ist. Wenn Sie mit der rechten Maustaste auf ein PortType-Kästchen klicken (entweder auf PortType, Operation, Message oder Message Part), erscheint ein Kontextmenü, in dem Sie die entsprechende Aktion auswählen können. Wenn Sie z.B. mit der rechten Maustaste auf einen PortType-Namen klicken, können Sie unter anderem einen neuen PortType oder eine Operation an den ausgewählten PortType anhängen oder ein Binding für den ausgewählten PortType erstellen. Beachten Sie: Wenn eine Komponente ausgewählt ist, können ihre Details in der Eingabehilfe "Details" bearbeitet werden.

Die Verknüpfung eines PortType mit einem Binding wird im Hauptfenster durch eine schwarze Konnektorlinie angezeigt, die das PortType-Kästchen mit dem Binding-Kästchen verbindet; das Binding-Kästchen befindet sich im Hauptfenster im Abschnitt "Bindings".

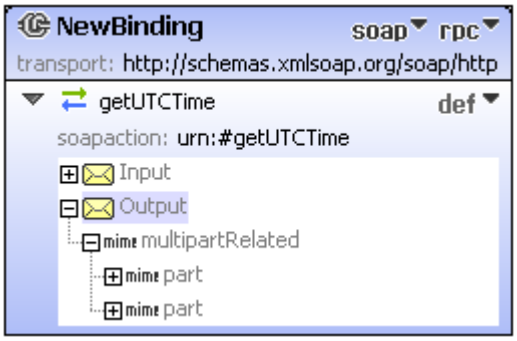
Bindings

In einem Binding sind das Message-Format und die Protokoll-Informationen für Operationen und Messages definiert, die für einen bestimmten PortType definiert sind. Bindings können für SOAP 1.1 Endpunkte und HTTP 1.1 GET und POST Verben erstellt werden. Jedes Binding wird im Hauptfenster im Abschnitt "Bindings" durch ein Binding-Kästchen dargestellt (*Abbildung unten*). Das Binding-Kästchen enthält alle Operationen des damit verknüpften PortType (*siehe Abbildung unten*).



Um das Binding zu definieren, wählen Sie in der ersten Auswahlliste rechts vom Namen des Binding (*Abbildung unten*) entweder `soap`, `http-get`, oder `http-post` aus, um die Art des Binding zu definieren. Bei Auswahl von `soap` können Sie (in der zweiten Auswahlliste) zusätzlich definieren, ob der Stil `doc` oder `rpc` sein soll.

MIME Encodings (auch als MIME-Bindings bezeichnet) werden auf Message-Ebene definiert. Klicken Sie zum Definieren eines MIME Encoding mit der rechten Maustaste auf die Message (*Abbildung unten*) und hängen Sie die entsprechende MIME-Definition an. In der Abbildung unten wurden MIME-Definitionen für die Message `Output` erstellt.

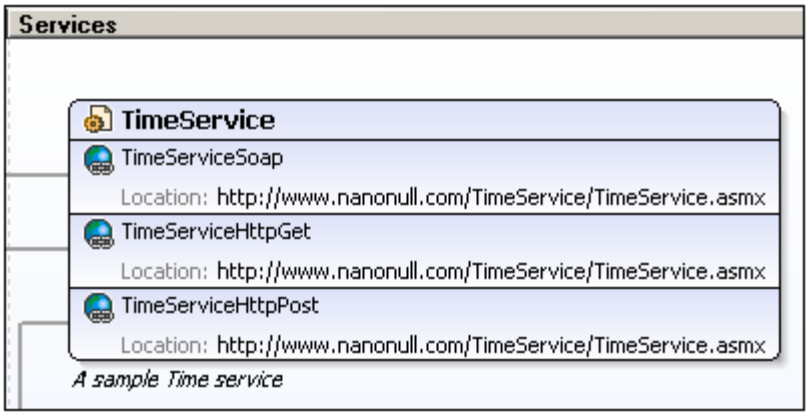


Wenn Sie mit der rechten Maustaste auf einen Eintrag in einem Binding-Kästchen klicken, wird ein kontextsensitives Menü angezeigt. Bindings können angehängt oder gelöscht werden; es können Extensibility-Elemente bearbeitet werden und Messages können definiert werden. Beachten Sie außerdem: Wenn ein Binding-Kästchen oder ein Eintrag in einem Binding-Kästchen ausgewählt ist, werden die Definitionen in der Eingabehilfe "Details" angezeigt und können dort bearbeitet werden.

Sie können für ein Binding einen Port erstellen. Klicken Sie dazu mit der rechten Maustaste auf die Titelleiste eines Binding-Kästchens und wählen Sie den Befehl "Port für Binding erstellen". Der dazugehörige Port wird (im Hauptfenster im Abschnitt "Services") in einem Service-Kästchen erstellt. Die Verknüpfung zwischen einem Binding und einem Port wird durch eine schwarze Konnektorlinie angezeigt.

Services
In einem Service sind eine Gruppe von miteinander in Zusammenhang stehenden Ports

zusammengruppiert. Es wird im Hauptfenster im Abschnitt "Services" in Form eines Service-Kästchens dargestellt (*Abbildung unten*. Jedes Service-Kästchen besteht aus einer oder mehreren Port-Deklarationen (*siehe Abbildung unten*).



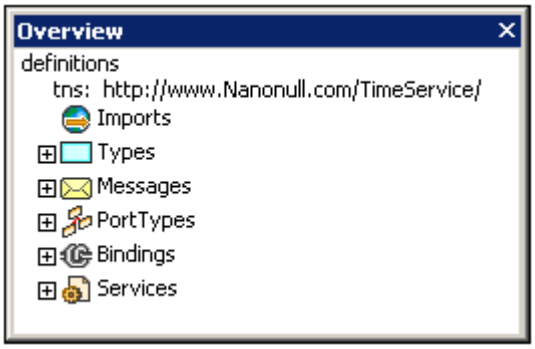
Der Name des Service, der Port-Name, das mit dem Port verknüpfte Binding und die Adressinformationen zu einem Port können direkt im Service-Kästchen oder in der Eingabehilfe "Details" bearbeitet werden. Wenn Sie mit der rechten Maustaste auf ein Service-Kästchen oder einen bestimmten Eintrag im Service-Kästchen klicken, wird ein Kontextmenü mit Befehlen zu diesem Service oder Eintrag geöffnet.

5.2.2 Eingabehilfen

Zur Bearbeitung von WSDL-Dokumenten stehen zwei Eingabehilfen zur Verfügung: Übersicht und Detail. Beide Eingabehilfen können durch Doppelklick auf die Titelleiste an- und abgedockt werden. Bei angedockter Eingabehilfe kann die Funktion zum automatischen Ausblenden aktiviert werden, indem Sie in der Titelleiste auf das Pin-Symbol klicken. Wenn die Eingabehilfe automatisch ausgeblendet wird, wird sie minimiert als Register am Rand des Applikationsfensters angezeigt. Eine automatisch ausgeblendete Eingabehilfe kann wieder angedockt werden, indem Sie sie vom Rand her ausrollen (indem Sie die Maus über das Register platzieren) und in der Titelleiste auf das Pin-Symbol klicken.

Eingabehilfe "Übersicht"

In der Eingabehilfe "Übersicht" (*Abbildung unten*) sehen Sie eine Übersicht über das WSDL-Dokument, wobei die verschiedenen Komponenten des Dokuments in Strukturkategorieren gruppiert sind und die Target Namespaces und die importierten Schemas und WSDL-Dokumente aufgelistetet werden. Zusätzlich zu PortTypes, Messages, Bindings und Services werden auch die verschiedenen im Dokument definierten Typen aufgelistet.



In jeder Kategorie werden die Komponenten in einer Baumstruktur angezeigt. Importierte Schemas und importierte WSDL-Dokumente werden unter der Überschrift "Imports" aufgelistet. Sie können die Einträge in der Baumstruktur erweitern und reduzieren um deren Inhalt anzuzeigen bzw. auszublenden. Wenn eine Komponente in der Eingabehilfe "Übersicht" ausgewählt wird, werden die Komponente und ihre Eigenschaften in der Eingabehilfe "Details" angezeigt, wo die Eigenschaften bearbeitet werden können. Beachten Sie die folgenden Punkte zu den einzelnen Kategorien in der Eingabehilfe "Übersicht".

Target Namespace: Wird in der Baumstruktur durch **`tns`** bezeichnet. Der Target Namespace kann in der Eingabehilfe "Übersicht" bearbeitet werden. Alle anderen Namespaces müssen in der Text-Ansicht bearbeitet werden.,

Imports: XML-Schema (XSD)-Dateien und WSDL-Dateien können in das aktive Dokument importiert werden. Um eine Datei zu importieren, rechtsklicken Sie auf den Eintrag "Imports" oder eine bereits importierte Datei in der Liste "Imports" und wählen Sie den Befehl "Add new import". Wenn Sie mit der rechten Maustaste in der Liste "Imports" auf eine bereits importierte Datei klicken, erscheint ein Kontextmenü, in dem Sie wählen können, ob Sie eine neue importierte Datei hinzufügen möchten, die importierte Datei durch eine andere Datei ersetzen möchten (Edit import) oder die importierte Datei löschen möchten. Außerdem können Sie die Datei von dem angegebenen Pfad aus auch öffnen. Die Datei wird in der Schema/WSDL Design-Ansicht geöffnet und kann dort editiert werden.

Types: Hier sind alle im WSDL-Dokument (schwarz) und in allen importierten Schemas oder importierten WSDL-Dokumenten (grau) definierten Typen aufgelistet. Durch Rechtsklick auf einen Typ in die Liste und Auswahl des entsprechenden "Add new"-Befehls können Sie ein neues XSD-Element, einen neuen simpleType oder complexType erstellen. Der neu erstellte Typ wird als Typ zu den WSDL `types` Definitionen hinzugefügt und in der Eingabehilfe "Übersicht" unter dem Eintrag "Types" aufgelistet. Ein im WSDL-Dokument definierter Typ kann in der Schema-Ansicht von XMLSpy bearbeitet werden. Rechtsklicken Sie dazu auf den gewünschten Typ und wählen Sie im Kontextmenü, das daraufhin angezeigt wird, "Typ/Element bearbeiten (Schema-Ansicht)" Daraufhin wird anhand der `types` Definitionen im WSDL-Dokument eine temporäre XSL-Datei on-the-fly generiert. Dieses XSD-Dokument wird in der Schema-Ansicht angezeigt und kann bearbeitet werden. Nachdem Sie die Bearbeitung des XSD-Dokuments abgeschlossen haben, werden die Änderungen beim Speichern wieder in den `types` Definitionen im WSDL-Dokument gespeichert. Sie können das XSD-Dokument auch schließen, ohne die Änderungen zu speichern; in diesem Fall werden die `types` Definitionen im WSDL-Dokument nicht geändert.

Messages: Wenn eine Message oder ihre Unterkomponente ausgewählt ist, werden die Eigenschaften der Message bzw. Unterkomponente in der Eingabehilfe "Detail" angezeigt, wo sie auch bearbeitet werden können. Wenn eine Message in der Eingabehilfe "Übersicht" ausgewählt ist, können Sie zusätzlich einen Message Part zu der Message hinzufügen, die Message löschen oder eine neue Message hinzufügen. Wenn in der Eingabehilfe "Übersicht" ein Message Part ausgewählt ist, können Sie einen weiteren Message Part zu dieser Message hinzufügen oder den ausgewählten Message Part löschen. Mit dem Befehl "Synchronisieren" wird die ausgewählte Message bzw. der ausgewählte Message Part im jeweiligen PortType-Kästchen markiert.

PortTypes: Wenn ein PortType ausgewählt ist, können PortTypes hinzugefügt werden, der ausgewählte PortType kann gelöscht werden und Operationen können zur Definition des ausgewählten PortType hinzugefügt werden. Wenn eine Operation ausgewählt ist, können zusätzliche Operationen angehängt werden, die ausgewählte Operation kann gelöscht werden und Elemente können zur ausgewählten Operation hinzugefügt werden. Wenn eine Message ausgewählt ist, können zusätzliche Messages hinzugefügt werden und die ausgewählte Message kann gelöscht werden. Mit dem Befehl "Synchronisieren" wird der ausgewählte PortType, die ausgewählte Operation bzw. die ausgewählte Message markiert.

 © 2007 Altova GmbH

Bindings: Wenn ein Binding ausgewählt ist, können zusätzliche Bindings an die bereits vorhandenen Bindings angehängt werden, das ausgewählte Binding kann gelöscht werden und Operationen können für das ausgewählte Binding eingefügt werden. Wenn eine Operation oder eine Message ausgewählt ist, stehen dieselben Optionen wie für Operationen und Messages in der Kategorie "PortTypes" zur Verfügung. Wenn Sie auf den Befehl "Synchronisieren" klicken, wird das ausgewählte Binding, die ausgewählte Operation oder Message markiert.

Services: Wenn ein Service ausgewählt ist, können zusätzliche Services hinzugefügt werden, der ausgewählte Service kann gelöscht werden und Ports können für den ausgewählten Service hinzugefügt werden. Wenn Sie auf den Befehl "Synchronisieren" klicken, wird der ausgewählte Service bzw. der ausgewählte Port markiert.

Eingabehilfe "Detail"
In der Eingabehilfe "Detail" werden die Eigenschaften des im Hauptfenster oder in der Eingabehilfe "Übersicht" ausgewählten Eintrags angezeigt (*Abbildung unten*). Diese Eigenschaften können in der Eingabehilfe "Detail" bearbeitet werden.

Wenn z.B., wie in der Abbildung oben gezeigt, der Port `TimeServiceSoap` (des `TimeService` Service) ausgewählt wird, so werden die Eigenschaften von `TimeServiceSoap` in der Eingabehilfe "Detail" angezeigt, wo sie auch bearbeitet werden können. Um ein Textfeld wie z.B. `name` oder `description` zu bearbeiten, doppelklicken Sie in das Feld und bearbeiten Sie den Text. Bei einigen Eigenschaften wie z.B. `binding` in der Abbildung oben, wo man aus mehreren Optionen auswählen kann, können Sie die gewünschte Option aus einer Auswahlliste auswählen.

6 Authentic-Ansicht

Die Authentic-Ansicht (siehe Abbildung unten) ist eine grafische Darstellung Ihres XML-Dokuments. Sie können damit XML-Dokumente **ohne** Markup-Code und **mit** den entsprechenden Formatierungs- und Dateneingabefunktionen wie z.B. Eingabefeldern, Listenfeldern und Optionsfeldern darstellen. Die Daten, die der Benutzer in der Authentic -Ansicht eingibt, werden in die XML-Datei eingegeben.

In der Authentic-Ansicht lassen sich XML-Dokumente und Datenbanken (DB) über eine einfache Benutzeroberfläche problemlos editieren.

Um ein XML-Dokument in der Authentic-Ansicht anzeigen und bearbeiten zu können, muss es mit einem **StyleVision Power Stylesheet** verknüpft werden, das in Altova <%SPY-STYLE%> erstellt wird. Ein StyleVision Power Stylesheet (. `sps`-Datei) ist im Grunde ein XSLT Stylesheet. Es definiert eine Ausgabedarstellung für ein XML-Dokument, das auch Dateneingabemechanismen enthält. Benutzer der Authentic-Ansicht können daher Daten zurück in die XML-Datei oder die DB schreiben. Ein StyleVision Power Stylesheet basiert auf einem Schema und ist schemaspezifisch. Wenn Sie ein StyleVision Power Stylesheet verwenden möchten, um eine XML-Datei in der Authentic-Ansicht zu editieren, muss es sich dabei um ein StyleVision Power Stylesheet handeln, das auf demselben Schema basiert, wie dasjenige, auf dem die XML-Datei basiert.

Verwendung der Authentic-Ansicht

- Wenn eine XML-Datei geöffnet ist, können Sie durch Klicken auf die Schaltfläche **Authentic** am unteren Rand des Dokumentenfensters in die Authentic-Ansicht wechseln. Wenn der XML-Datei noch kein StyleVision Power Stylesheet zugewiesen wurde, werden Sie aufgefordert, dies jetzt zu tun. Es muss ein StyleVision Power Stylesheet verwendet werden, das auf demselben Schema basiert, wie die XML-Datei.

 © 2007 Altova GmbH

- Wenn Sie den Befehl **Datei | Neu** wählen und anschließend auf die Schaltfläche " StyleVision Stylesheet auswählen..." klicken, wird in der Authentic-Ansicht eine neue XML-Datei erstellt und angezeigt. Diese neue Datei ist eine Vorlagendatei, die mit dem StyleVision Power Stylesheet verknüpft ist, das Sie öffnen. Sie kann bereits verschiedene Ausgangsdaten enthalten. Diese Ausgangsdaten befinden sich in einer XML-Datei (einer XML-Vorlagendatei), die optional mit dem StyleVision Power Stylesheet verknüpft werden kann. Nachdem die Authentic-Ansicht einer XML-Datei nun angezeigt wird, können Sie Daten darin eingeben und die Datei speichern.

Dieser Abschnitt enthält:

- eine Übersicht über die Benutzeroberfläche
- eine Beschreibung der Symbolleistensymbole der Authentic-Ansicht
- eine Beschreibung der Ansichtsmodi im Authentic-Ansicht-Hauptfenster
- eine Beschreibung der Eingabehilfen und deren Verwendung
- eine Beschreibung der verschiedenen in der Authentic-Ansicht verfügbaren Kontextmenüs
- eine Erläuterung zur Verwendung der verschiedenen Funktionen der Authentic-Ansicht

Weitere Informationen zur Authentic-Ansicht finden Sie:

- im Tutorial zur Authentic-Ansicht, in dem beschrieben wird, wie Sie mit der Authentic -Ansicht arbeiten. Sie finden das Tutorial in der Dokumentation zu Altova XMLSpy und zur Altova Authentic Desktop (siehe Abschnitt Tutorials) sowie online.
- Eine ausführliche Beschreibung der Menübefehle der Authentic-Ansicht finden Sie im Abschnitt "Benutzerreferenz" Ihrer Produktdokumentation.

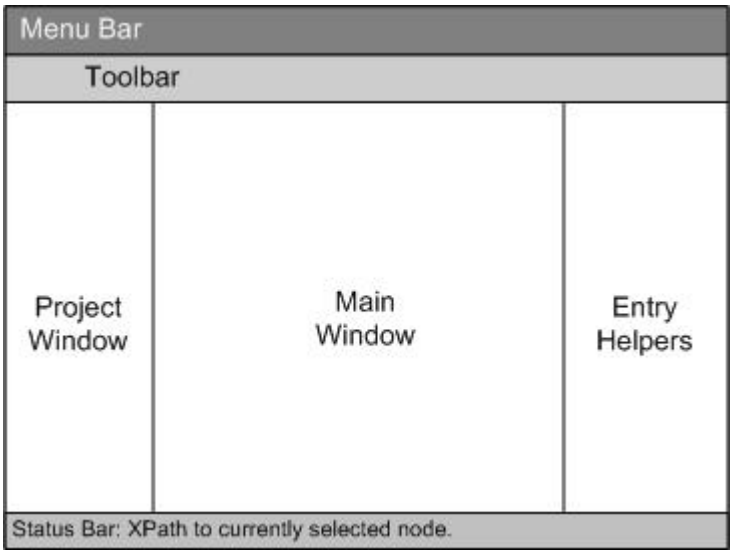
6.1 Benutzeroberfläche

In diesem Abschnitt wird die Benutzeroberfläche der Authentic-Ansicht beschrieben. Sie finden darin die folgenden Unterabschnitte:

- Übersicht über die Benutzeroberfläche
- Authentic-Ansicht Symbolleistenschaltflächen
- Authentic-Ansicht Hauptfenster
- Authentic-Ansicht Eingabehilfen
- Authentic-Ansicht Kontextmenüs

6.1.1 Übersicht über die Benutzeroberfläche

Die Authentic-Ansicht besteht aus einer Menüleiste am oberen Rand des Fensters und drei Bereichen: dem Projekt-Fenster, dem Hauptfenster und den Eingabehilfen-Fenstern (siehe Abbildung unten).



Menüleiste
Die Menüs der Menüleiste werden in der Produktdokumentation im Abschnitt "Benutzerreferenz" näher beschrieben.

Symbolleiste
Eine Beschreibung der Schaltflächen der Symbolleiste finden Sie im Abschnitt Symbolleistenschaltflächen der Authentic-Ansicht.

Projekt-Fenster
XML-, XSL-, HTML-Schema und Entity-Dateien können in einem Projekt zusammen gruppiert werden. Verwenden Sie zum Erstellen und Bearbeiten der Liste der Projektdateien die Befehle im Menü "Projekt" (Beschreibung siehe Abschnitt "Benutzerreferenz" der Produktdokumentation). Die Liste der Projektdateien wird im Projekt-Fenster angezeigt. Um eine Datei im Projekt-Fenster aufzurufen, doppelklicken Sie darauf.

Hauptfenster
Das Hauptfenster ist der Bereich, in dem das XML-Dokument angezeigt und bearbeitet wird. Eine Beschreibung dazu finden Sie im Abschnitt Authentic-Ansicht Hauptfenster.

Eingabehilfen

© 2007 Altova GmbH

Dieser Bereich enthält drei Eingabehilfen-Fenster: Elemente, Attribute und Entities. Welche Einträge in diesen Fenstern (Element- und Attribute-Eingabehilfen) zu sehen sind, ist vom Kontext abhängig, d.h. davon, an welcher Stelle im Dokument sich der Cursor gerade befindet. Um ein Element oder eine Entity in das Dokument einzugeben, doppelklicken Sie in der Eingabehilfe darauf. Der Wert eines Attributs wird in der Attribute-Eingabehilfe im Feld "Wert" des entsprechenden Attributs eingegeben. Nähere Informationen dazu finden Sie im Abschnitt Authentic-Ansicht Eingabehilfen.

Statusleiste
In der Statusleiste wird der XPath zum aktuell ausgewählten Node angezeigt.

Kontextmenüs
Dies sind Menüs, die angezeigt werden, wenn Sie im Hauptfenster auf die rechte Maustaste klicken. Es stehen Ihnen kontextsensitive Editierbefehle zur Verfügung, mit Hilfe derer Sie die Struktur und den Inhalt des ausgewählten Nodes bearbeiten können. Diese Befehle gestatten das Einfügen, Anhängen oder Löschen eines Nodes, das Hinzufügen von Entities und das Ausschneiden und Einfügen von Inhalt.

6.1.2 Authentic-Ansicht Symbolleistenschaltflächen

Die Schaltflächen der Symbolleiste der Authentic-Ansicht sind Befehls-Shortcuts. Einige Symbole kennen Sie sicher bereits von anderen Windows-Anwendungen oder anderen Altova-Produkten her, andere sind vielleicht neu für Sie. In diesem Abschnitt werden die für die Authentic-Ansicht spezifischen Befehle beschrieben.

In der nachfolgenden Beschreibung sind verwandte Befehle gemeinsam gruppiert.

Wechseln in die Authentic-Ansicht

Wenn das XML-Dokument mit einem StyleVision Power Stylesheet verknüpft ist, wechseln Sie durch Auswahl des Befehls **Ansicht | Authentic** aus einer anderen Ansicht in die Authentic-Ansicht.
Wenn das Dokument **nicht** mit einem StyleVision Power Stylesheet verknüpft ist, wird ein Dialogfeld angezeigt, in dem Sie aufgefordert werden, das Dokument mit einem StyleVision Power Stylesheet zu verknüpfen. Falls Sie eine Meldung erhalten, dass keine temporäre Datei (Temp-Datei) erstellt werden konnte, wenn Sie in die Authentic-Ansicht zu wechseln versuchen, wenden Sie sich an Ihren Systemadministrator. Der Systemadministrator muss die Standard-Sicherheits-ID für "non-power User" ändern, damit diesen das Anlegen von Ordnern und Dateien gestattet ist.

Ein- und Ausblenden von XML Markup
In der Authentic-Ansicht können die Tags für alle, einige oder keines der XML-Elemente oder Attribute eingeblendet werden - entweder mit Namen (große Markup-Symbole) oder ohne Namen (kleine Markup-Symbole). Die vier Markup-Symbole werden in der Symbolleiste angezeigt, die entsprechenden Befehle finden Sie im Menü **Authentic.**

Markup-Symbole ausblenden. Alle XML-Tags mit Ausnahme der reduzierten Tags werden ausgeblendet. Wenn Sie im Modus "Markup-Symbole ausblenden" auf einen Tag, der reduziert wurde, doppelklicken (wodurch er normalerweise erweitert wird), wird der Inhalt des Nodes angezeigt und die Tags ausgeblendet.

Kleine Markup-Symbole einblenden. Die XML-Element/Attribut-Tags werden ohne Namen angezeigt.

Große Markup-Symbole einblenden. Die XML-Elemente/Attribute werden mit ihren Namen angezeigt.

Mixed Markup-Symbole einblenden. Im StyleVision Power Stylesheet können Sie für jedes XML-Element oder Attribut einzeln festlegen, wie es im "Mixed Markup"-Modus angezeigt werden soll (mit großen oder kleinen Markup-Symbolen oder gar nicht). Im Mixed Markup-Modus der Authentic-Ansicht ist die Anzeige der Markup-Symbole also benutzerdefiniert. Beachten Sie allerdings, dass diese Einstellungen von der Person vorgenommen werden, die das StyleVision Power Stylesheet erstellt hat.

Bearbeiten von dynamischen Tabellenstrukturen
Zeilen in einer **dynamischen SPS-Tabelle** sind Wiederholungen einer Datenstruktur. Jede Zeile steht für eine Instanz eines einzelnen Elements, daher hat jede Zeile dieselbe XML-Substruktur wie die darauf folgende.

Die Befehle zum Bearbeiten dynamischer Tabellen dienen zum Bearbeiten der Zeilen einer dynamischen SPS-Tabelle, d.h. Sie können die Anzahl und Reihenfolge der Element-Instanzen ändern. Sie können jedoch nicht die Spalten einer dynamischen SPS-Tabelle bearbeiten, da dadurch die Substruktur der einzelnen Elementinstanzen geändert würde.

Die Symbole zum Bearbeiten dynamischer Tabellen werden in der Symbolleiste angezeigt und können auch über das Menü **Authentic** aufgerufen werden.

Zeile anhängen

Zeile einfügen

Zeile kopieren (d.h. der Inhalt einer Zelle wird dupliziert)

Zeile nach oben

Zeile nach unten

Zeile löschen

Bitte beachten Sie: Diese Befehle sollten nur bei **dynamischen SPS-Tabellen** und nicht innerhalb von statischen SPS-Tabellen verwendet werden. Die verschiedenen Arten der in der Authentic-Ansicht verwendeten Tabellen werden weiter hinten im Abschnitt Tabellen in der Authentic-Ansicht näher beschrieben.

Erstellen und Bearbeiten von XML-Tabellen
Sie können Ihre eigenen Tabellen einfügen, wenn Sie Daten in Tabellenform darstellen möchten. Solche Tabellen werden als XML-Tabellen eingefügt. Sie können die Struktur einer XML-Tabelle bearbeiten und die Tabelle formatieren. Die Symbole dazu stehen in der

Symbolleiste zur Verfügung (siehe Abbildung unten). Im Abschnitt XML-Tabellenbearbeitungsbefehle finden Sie nähere Informationen zu diesen Befehlen.

Die Befehle zu diesen Symbolen **stehen nicht als Menübefehle** zur Verfügung. Beachten Sie außerdem, dass die entsprechende Funktion erst im StyleVision Power Stylesheet aktiviert und konfiguriert werden muss, damit Sie XML-Tabellen verwenden können.

Eine ausführliche Beschreibung der in der Authentic-Ansicht verwendeten Tabellen bzw. wie diese erstellt und bearbeitet werden, finden Sie weiter hinten in der Dokumentation im Abschnitt Tabellen in der Authentic-Ansicht.

Symbole zur Textformatierung
Text wird in der Authentic-Ansicht formatiert, indem man darauf ein XML-Element oder -Attribut anwendet, das die gewünschte Formatierung aufweist. Wurde eine solche Formatierung definiert, kann der Autor des StyleVision Power Stylesheet in der Authentic-Ansicht Symbole zur Verfügung stellen, mit denen die Formatierung angewendet wird.

Um eine Textformatierung mit Hilfe eines Textformatierungssymbols anzuwenden, markieren Sie den gewünschten Text und klicken Sie auf das entsprechende Symbol.

Symbole für die Navigation in DB-Zeilen

Die Pfeilsymbole von links nach rechts sind: Gehe zum ersten Datensatz in der DB; Gehe zum vorhergehenden Datensatz; Dialogfeld "Gehe zu Datensatz #"; Gehe zum nächsten Datensatz; und Gehe zum letzten Datensatz.

Mit diesem Symbol wird das Dialogfeld "Datenbankabfrage bearbeiten" aufgerufen. Hier können Sie eine Abfrage eingeben. In der Authentic-Ansicht werden die abgefragten Datensätze angezeigt.

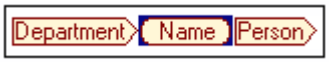
6.1.3 Authentic-Ansicht Hauptfenster

In der Authentic-Ansicht gibt es vier Ansichtsmodi: Große Markup-Symbole; Kleine Markup-Symbole; Mixed Markup-Symbole; und Alle Markup-Symbole ausblenden. Damit können Sie die Markup-Informationen für Dokumente mit unterschiedlicher Genauigkeit anzeigen.

Um zwischen diesen Ansichtsmodi zu wechseln, verwenden Sie die Befehle im Menü "Authentic" oder die Symbole der Symbolleiste (siehe vorhergehender Abschnitt, Authentic-Ansicht Symbolleistenschaltflächen).

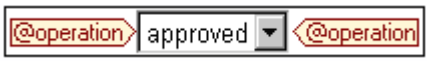
Große Markup-Symbole
In diesem Modus werden die Start- und End-Tags von Elementen und Attributen mit dem Namen des Elements/Attributs im Tag angezeigt:

Das Element `Name` in der Abbildung oben ist **erweitert**, d.h. sein Start- und End-Tag sowie der Inhalt der Elements werden angezeigt. Durch Doppelklicken auf den Start- oder End-Tag eines Elements/Attributs können Sie dieses **reduzieren.**

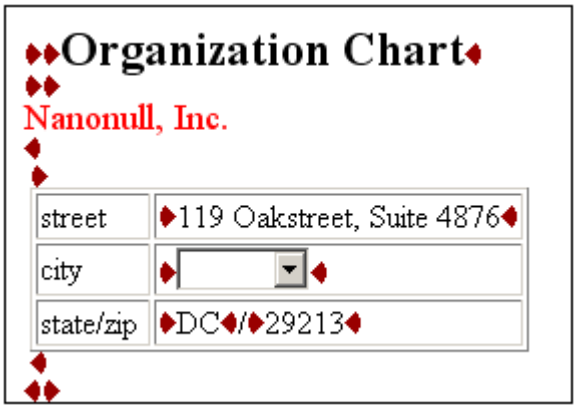


Um das reduzierte Element/Attribut zu erweitern, doppelklicken Sie auf den reduzierten Tag.

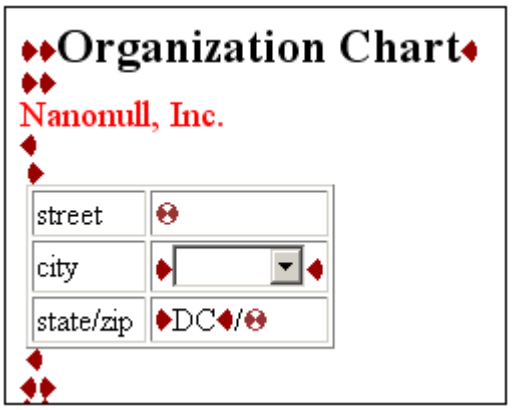
Im Modus für große Markup-Symbole werden Attribute durch das Symbol @ im Start- und End-Tag gekennzeichnet:



Kleine Markup-Symbole
In diesem Modus werden die Start- und End-Tags von Elementen/Attributen ohne Namen angezeigt:



Um ein Element/Attribut zu reduzieren bzw. zu erweitern, doppelklicken Sie auf den entsprechenden Tag. Im unten gezeigten Beispiel sehen Sie zwei reduzierte Elemente in der Tabelle:



Mixed Markup-Symbole
Im Mixed Markup-Modus wird eine benutzerdefinierte Markup-Ansicht angezeigt. Der Autor des StyleVision Power Stylesheet kann für einzelne Elemente/Attribute im Dokument große, kleine oder gar keine Markup-Symbole einstellen. Der Benutzer der Authentic-Ansicht sieht diese benutzerdefinierte Ansicht im "Mixed Markup"-Ansichtsmodus.

Alle Markup-Symbole ausblenden
In diesem Modus werden alle XML-Markup-Symbole ausgeblendet. Da die Formatierung in der Authentic-Ansicht die Formatierung des gedruckten Dokuments ist, handelt es sich bei diesem Modus um eine WYSIWYG-Ansicht des Dokuments.

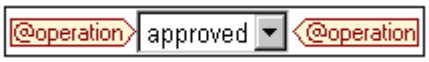
© 2007 Altova GmbH

Anzeige von Inhalt
In der Authentic-Ansicht wird Inhalt auf zwei Arten angezeigt:

- Reiner Text. Sie geben den Text ein und dieser Text wird der Inhalt des Elements bzw. der Wert des Attributs.

- Dateneingabe-Hilfen. Dabei handelt es sich entweder um ein Eingabefeld (Textfeld), ein Eingabefeld mit mehreren Zeilen, eine Dropdown-Liste, ein Kontrollkästchen oder ein Optionsfeld. Text, den Sie in Eingabefelder und Eingabefelder mit mehreren Zeilen eingeben, wird der XML-Inhalt des Elements bzw. der Wert des Attributs.

Bei den anderen Dateneingabe-Hilfen wird durch Ihre Auswahl der entsprechende XML-Wert erzeugt, der im StyleVision Power Stylesheet definiert wurde. Wenn also im Beispiel unten "approved" ausgewählt wird, wird dieser Eintrag auf einen XML-Wert "1" oder auf "approved" oder einen beliebigen anderen Eintrag gemappt, während "not approved" dem Wert "0" oder "not approved" oder einem beliebigen anderen Wert zugeordnet wird.

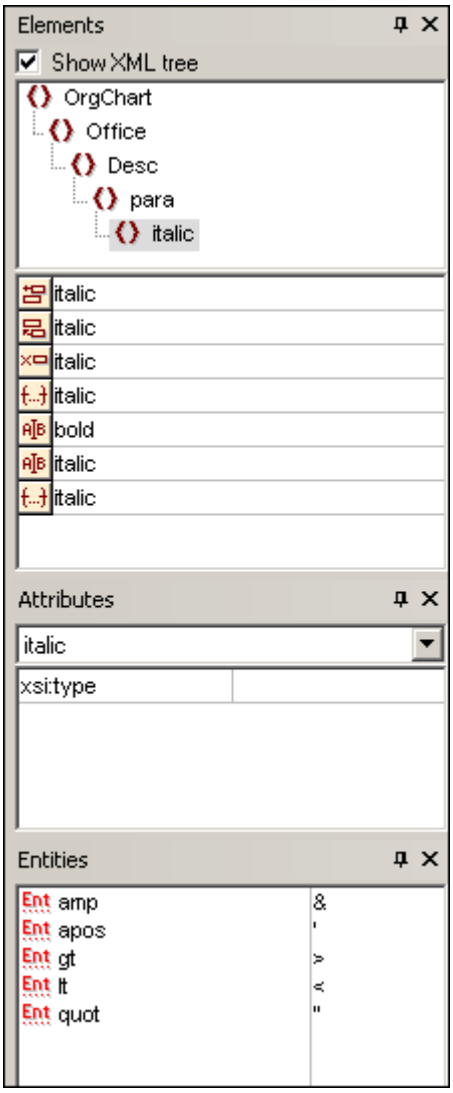


Optionale Nodes
Wenn ein Element oder Attribut gemäß dem referenzierten Schema **optional** ist, wird eine Eingabeaufforderung vom Typ "add [*Element/Attribut*]" angezeigt.

Wenn Sie auf die Eingabeaufforderung klicken, wird das Element hinzugefügt und der Cursor so gesetzt, dass Daten eingegeben werden können. Wenn es mehrere optionale Nodes gibt, wird die Aufforderung "add..." angezeigt. Wenn Sie darauf klicken, wird ein Menü mit den optionalen Nodes angezeigt.

6.1.4 Authentic-Ansicht Eingabehilfen

Die Authentic-Ansicht verfügt über drei Eingabehilfen: für Elemente, Attribute und Entities. Sie werden als Fenster am rechten Rand der Authentic-Ansicht angezeigt.



Die Element- und Attribut-Eingabehilfen sind kontextsensitiv, d.h. was in der Eingabehilfe angezeigt wird, hängt davon ab, an welcher Stelle im Dokument der Cursor sich befindet. Die in der Entities-Eingabehilfe angezeigten Entities sind nicht kontextsensitiv; es werden - unabhängig von der Cursorposition - alle im Dokument zulässigen Entities angezeigt.

Im Folgenden werden die einzelnen Eingabehilfen beschrieben.

Element-Eingabehilfe
Die Element-Eingabehilfe besteht aus zwei Teilen:

- dem oberen Teil mit einer XML-Baumstruktur, der über das Kontrollkästchen **Show xml tree** ein- und ausgeblendet werden kann. In der XML-Baumstruktur sehen Sie die übergeordneten Nodes bis hinauf zum Root-Element des aktuellen Elements des Dokuments. Wenn Sie auf ein Element in der XML-Baumstruktur klicken, werden die dazugehörigen Elemente (wie in nächsten Punkt dieser Liste beschrieben) im unteren Teil der Element-Eingabehilfe angezeigt .
- Im unteren Teil der Element-Eingabehilfe werden die Elemente aufgelistet, die im, vor und nach dem ausgewählten Element/Textbereich in der Authentic-Ansicht eingefügt werden können, entfernt werden können oder darauf (d.h. durch Ersetzung) angewendet werden können. Was Sie mit einem in der Eingabehilfe aufgelisteten Element tun können, wird in der

© 2007 Altova GmbH

Eingabehilfe durch das Symbol auf der linken Seite des Elementnamens angezeigt. Im Folgenden finden Sie eine Liste der in der Element-Eingabehilfe vorkommenden Symbole mit einer Erklärung dazu.

Um einen Node aus der Eingabehilfe zu verwenden, klicken Sie auf sein Symbol.

Hinter Element einfügen
Das Element in der Eingabehilfe wird hinter dem ausgewählten Element eingefügt. Beachten Sie, dass es auf der richtigen hierarchischen Ebene angehängt wird. Wenn sich Ihr Cursor z.B. innerhalb eines `//sect1/para` -Elements befindet und Sie ein `sect1`-Element anhängen, wird das neue `sect1`-Element nicht als Geschwisterelement von `//sect1/para` angehängt, sondern als `sect1`-Element, das ein Parent dieses `para`-Elements ist.

Vor Element einfügen
Das Element in der Eingabehilfe wird vor dem ausgewählten Element eingefügt. Beachten Sie, dass das Element genau wie beim vorhergehenden Befehl auf der richtigen hierarchischen Ebene eingefügt wird.

Element entfernen
Löscht des Element und dessen Inhalt.

Element einfügen
Sie können ein Element aus der Eingabehilfe auch **innerhalb** eines Elements einfügen. Wenn Sie den Cursor in ein Element hineinsetzen, können die zulässigen Child-Elemente dieses Elements eingefügt werden. Beachten Sie, dass zulässige Child-Elemente sowohl Teil eines "elements-only Content Model" als auch Teil eines "Mixed Content Model" (Text plus Child-Elemente) sein können.

Ein zulässiges Child-Element kann entweder dann eingefügt werden, wenn ein Textbereich markiert wurde oder wenn der Cursor als Einfügepunkt in den Text platziert wurde.

- Wenn ein Textbereich ausgewählt und ein Element eingefügt wurde, wird der Textbereich der Inhalt des eingefügten Elements.
- Bei Auswahl einer Cursorposition wird das Element an dieser Stelle eingefügt

Nachdem das Element eingefügt wurde, kann es durch Klicken auf eines der beiden Symbole zum Löschen eines Elements (in der Element-Eingabehilfe) gelöscht werden. Welches der beiden Symbole angezeigt wird, ist abhängig davon, ob Sie einen Textbereich auswählen oder ob Sie den Cursor als Einfügepunkt in den Text setzen (siehe unten).

Element anwenden
Wenn Sie in Ihrem Dokument ein Element auswählen (indem Sie in der "Große Markup-Symbol anzeigen"-Ansicht entweder auf den Start- oder den End-Tag eines Elements klicken) und wenn dieses Element durch ein anderes Element ersetzt werden kann (z.B. kann in einem Element gemischten Inhalts wie `para`, ein `italic` Element durch das `bold` Element ersetzt werden), bedeutet dieses Symbol, dass das Element in der Eingabehilfe auf das ausgewählte (Original-) Element angewendet werden kann. Der Befehl **Element anwenden** kann auch auf einen Textbereich innerhalb eines Elements gemischten Inhalts angewendet werden; der Textbereich wird als Inhalt des angewendeten Elements erstellt.

- Wenn das angewendete Element ein **Child-Element mit demselben Namen** wie das Child-Element des Originalelements hat und das Original-Element eine Instanz dieses Child-Elements enthält, dann wird das Child-Element des Originals im neuen Element-Inhalt beibehalten.
- Wenn das angewendete Element **kein Child-Element mit demselben Namen** wie das instantiierte Child-Element des Originalelements enthält, dann wird das instantiierte Child-Element des Originalelements als Geschwisterelement von etwaigen Child-Elementen oder Elementen des neuen Elements angehängt.
- Wenn das angewendete Element **ein Child-Element hat, für das es im Content Model des Originalelements kein Äquivalent gibt**, wird dieses Child-Element nicht direkt erstellt, doch bietet Ihnen die Authentic-Ansicht die Möglichkeit, dieses einzufügen.

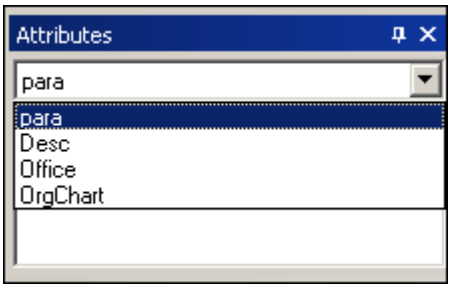
Wenn Sie anstatt eines Elements einen Textbereich auswählen, wird bei Anwendung eines Elements auf diesen Bereich an dieser Stelle das angewendete Element erstellt, wobei der markierte Text dessen Inhalt wird. Wenn der Cursor sich an einem Einfügepunkt befindet, ist die Anwendung eines Elements nicht zulässig.

Element löschen (wenn ein Bereich ausgewählt ist)
Dieses Symbol erscheint, wenn Text innerhalb eines Elements gemischten Inhalts ausgewählt wird. Wenn Sie auf dieses Symbol klicken, wird das Element, das den ausgewählten Bereich einschließt, gelöscht.

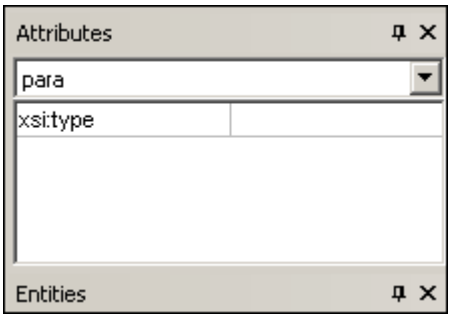
Element löschen (wenn ein Einfügepunkt ausgewählt ist)
Dieses Symbol erscheint, wenn der Cursor in ein Element platziert wird, das ein Child eines Elements gemischten Inhalts ist. Wenn Sie auf das Symbol klicken, wird dieses zeileninterne Element gelöscht.

Attribut-Eingabehilfe
Die Attribut-Eingabehilfe besteht aus einem Listenfeld mit einer Dropdown-Liste von Attributen. Das ausgewählte Element (Sie können auf den Start- oder End-Tag klicken oder den Cursor irgendwo im Element-Inhalt platzieren) wird in der Dropdown-Liste angezeigt.

Die in den nachfolgenden Abbildungen gezeigte Attribut-Eingabehilfe hat ein `para`-Element in der Dropdown-Liste. Wenn Sie auf den Abwärtspfeil in der Liste klicken, erscheint eine Liste aller **übergeordneten Elemente** des `para`-Elements hinauf bis zum Root-Element, in diesem Fall OrgChart.



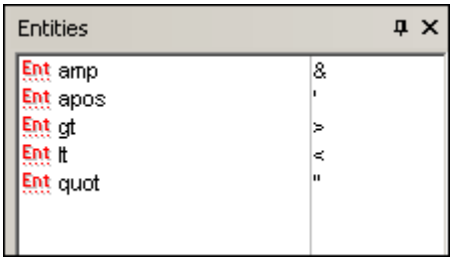
Unterhalb der Dropdown-Liste wird eine Liste gültiger Attribute für dieses Element - in diesem Fall für das `para`-Element angezeigt. Wenn ein Attribut für ein bestimmtes Element obligatorisch ist, wird es fett angezeigt. (Im Beispiel unten gibt es keine obligatorischen Attribute)



Um den Wert für ein Attribut einzugeben, klicken Sie in das Wertefeld des Attributs und geben Sie den Wert ein. Daraufhin wird das Attribut mit dem entsprechenden Wert im XML-Dokument erstellt.

Im Falle des Attributs `xsi:nil`, welches in der Eingabehilfe "Attribut" erscheint, wenn ein nillable Element ausgewählt wurde, kann der Wert des Attributs `xsi:nil` nur durch Auswahl eines der zulässigen Werte (`true` oder `false`) aus der Dropdown-Liste der Werte des Attributs eingegeben werden.

Entities-Eingabehilfe
Über die Entities-Eingabehilfe können Sie eine Entity in Ihr Dokument einfügen. Entities dienen zum Einfügen von Sonderzeichen oder Textfragmenten, die in einem Dokument öfter vorkommen (z.B. der Name einer Firma). Um eine Entity einzufügen, setzen Sie den Cursor an die gewünschte Stelle im Text und doppelklicken Sie in der Entities-Eingabehilfe auf die Entity.



Anmerkung:
Eine interne Entity ist eine Entitiy, deren Wert in der DTD definiert wurde. Eine externe Entity ist eine Entity, deren Wert in einer externen Quelle z.B. einer anderen XML-Datei enthalten ist. Sowohl interne als auch externe Entities werden in der Entities-Eingabehilfe aufgelistet. Beim Einfügen einer Entity - egal ob intern oder extern - wird die Entity und nicht ihr Wert in den XML-Text eingefügt. Wenn es sich um eine interne Entity handelt, wird in der Authentic-Ansicht **der Wert der Entity** angezeigt. Dies bedeutet, dass eine XML-Datei, die eine externe Entity ist, in der Authentic-Ansicht als Entity angezeigt wird; die Entity wird in der Authentic-Ansicht nicht durch ihren Inhalt ersetzt.

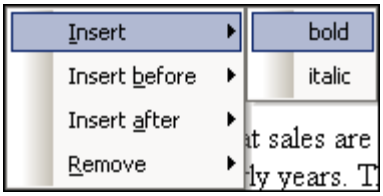
Sie können in der Authentic-Ansicht auch Ihre **eigenen Entities definieren**: Nähere Informationen dazu finden Sie im Abschnitt "Bearbeitung in der Authentic -Ansicht" unter Definieren von Entities.

6.1.5 Authentic-Ansicht Kontextmenüs

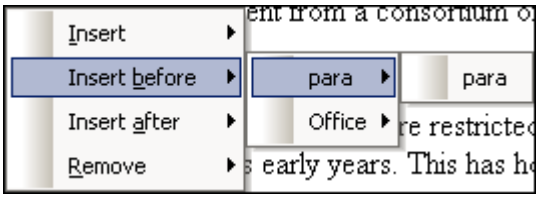
Wenn Sie mit der rechten Maustaste auf einen ausgewählten Dokumenteninhalt oder Node klicken, wird ein Kontextmenü mit Befehlen, die für die Auswahl oder Cursorposition relevant sind, angezeigt.

Einfügen von Elementen
In der Abbildung unten sehen Sie das Kontextmenü mit dem Untermenü **Einfügen**, das eine

Liste aller Elemente enthält, die an der aktuellen Cursorposition eingefügt werden können. Im Untermenü **Einfügen vor** werden alle Elemente aufgelistet, die vor dem aktuellen Element eingefügt werden können. Im Untermenü **Einfügen nach** werden alle Elemente aufgelistet, die nach dem aktuellen Element eingefügt werden können. In der Abbildung unten ist das aktuelle Element das `para`-Element. Die Elemente `italic` und `bold` können innerhalb des aktuellen `para`-Elements eingefügt werden.



Die Elemente `para` und `Office` können vor dem `para`-Element eingefügt werden.



Die meisten der Befehle aus dem Kontextmenü werden unter Authentic-Ansicht Eingabehilfen beschrieben.

Entfernen eines Node
Wenn Sie den Mauszeiger über den Befehl **Entfernen** positionieren, erscheint eine Menüliste bestehend aus dem ausgewählten Node und allen seinen übergeordneten Nodes bis hinauf zum Dokument-Element, die entfernt werden können (ohne dass das Dokument ungültig wird). Klicken Sie auf das Element, das Sie entfernen möchten. Auf diese Art lässt sich ein Element oder ein beliebiges übergeordnetes Element einfach entfernen. Beachten Sie, dass bei Entfernung eines übergeordneten Elements auch alle untergeordneten Elementen einschließlich des ausgewählten Elements entfernt werden.

Einfügen einer Entity
Wenn Sie den Cursor über den Befehl "Entity einfügen" positionieren, erscheint ein Untermenü mit einer Liste aller deklarierten Entities. Wenn Sie auf eine Entity klicken, wird sie am ausgewählten Node eingefügt. Eine Beschreibung, wie man Entities für das Dokument definiert, finden Sie unter Definieren von Entities.

Einfügen eines CDATA-Abschnitts
Dieser Befehl ist aktiviert, wenn Sie den Cursor innerhalb von Text setzen. Wenn Sie auf den Befehl klicken, wird am Cursor-Punkt ein CDATA-Abschnitt eingefügt. Der CDATA-Abschnitt steht innerhalb eines Start- und End-Tags; um diese Tags anzuzeigen, aktivieren Sie die großen oder kleinen Markup-Symbole. Innerhalb von CDATA-Abschnitten werden XML-Markup-Symbole und Parsing ignoriert. XML-Markup-Zeichen (Zeichen wie das kaufmännische Und, Apostroph, größer als, kleiner als und Anführungszeichen) werden nicht als Markup behandelt sondern als Literalzeichen. Daher eignen sich CDATA-Abschnitte für Text wie z.B. Auflistungen von Programmcode, welche XML-Markup-Zeichen enthaltent.

Entfernen
Mit dem Befehl "Entfernen" können Sie den ausgewählten Node und seinen Inhalt entfernen. Ein Node wird dann als ausgewählt betrachtet, wenn Sie den Cursor in den Node platzieren oder auf den Start- oder End-Tag des Node klicken.

 © 2007 Altova GmbH

Löschen
Mit dem Befehl "Löschen" wird der Element-Markup-Code rund um die Auswahl gelöscht. Wenn der gesamte Node ausgewählt ist, wird der Element-Markup-Code für den gesamten Node gelöscht. Wenn ein Textabschnitt ausgewählt ist, wird der Element-Markup-Code nur rund um diesen Textabschnitt gelöscht.

6.2 Bearbeitung in der Authentic-Ansicht

In diesem Abschnitt werden wichtige Features der Authentic-Ansicht näher beschrieben. Bei den Features, die hier behandelt werden, handelt es sich entweder um häufig gebrauchte Funktionen oder solche, deren Funktionsweise und Konzepte näher erklärt werden müssen.

In diesem Abschnitt werden folgende Themen behandelt:

- Es gibt in der Authentic-Ansicht drei verschiedene Tabellenarten. Im Abschnitt Tabellen in der Authentic-Ansicht werden diese drei Tabellenarten (statisches SPS, dynamisches SPS und XML) näher erläutert und deren Verwendung wird erklärt. Die Beschreibung beginnt mit einem allgemeinen Überblick. Anschließend wird näher auf die Einzelheiten eingegangen.
- Die Datumsauswahl ist ein grafischer Kalender, der Daten im korrekten XML-Format eingibt, wenn Sie auf ein Datum klicken. Siehe Datumsauswahl.
- Eine Entity ist ein Kürzel für ein Sonderzeichen oder einen Textstring. Sie können Ihre eigenen Entities definieren, um Sonderzeichen oder Textstrings durch Eingabe der Entity einfügen zu können. Nähere Informationen dazu finden Sie unter Definieren von Entities.
- Die Grafikformate, die in der Authentic-Ansicht angezeigt werden können.

6.2.1 Tabellen in der Authentic-Ansicht

Die drei Tabellentypen fallen in zwei Kategorien: SPS-Tabellen (statische und dynamische) und XML-Tabellen.

SPS-Tabellen: Es gibt zwei Arten von SPS-Tabellen: statische und dynamische. SPS-Tabellen werden vom Autor des StyleVision Power Stylesheet, mit dem Ihr XML-Dokument verknüpft ist, erstellt. Sie können selbst keine SPS-Tabelle in das XML-Dokument einfügen, doch können Sie Daten in die Felder der SPS-Tabelle eingeben und Zeilen in dynamischen SPS-Tabellen hinzufügen und löschen. Im nachfolgenden Abschnitt zu SPS-Tabellen werden die Funktionen dieser Tabellen näher beschrieben.

XML-Tabellen werden vom Benutzer der Authentic-Ansicht eingefügt. Auf diese Art können Sie auf Wunsch an jeder Stelle im Dokument, an der dies zulässig ist, Tabellen einfügen. Die Editierfeatures von XML-Tabellen und die Bearbeitungssymbole für XML-Tabellen werden im Folgenden beschrieben.

SPS-Tabellen

In der Authentic-Ansicht werden zwei Arten von SPS-Tabellen verwendet: statische Tabellen und dynamische Tabellen.

Statische Tabellen sind von der Struktur und der Art des Inhalts der Zellen unveränderbar. Als Benutzer der Authentic-Ansicht können Sie Daten in die Tabellenzellen eingeben, die Struktur der Tabellen (d.h. Hinzufügen von Zeilen oder Spalten usw.) oder den Inhaltstyp einer Zelle jedoch nicht verändern. Die Eingabe von Daten erfolgt entweder durch Eingabe von Text oder durch Auswahl von Optionen in Form von Kontrollkästchen, Optionsfeldern oder Dropdown-Listen. Nachdem Sie Daten eingegeben haben, können Sie diese bearbeiten.

© 2007 Altova GmbH

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

Bitte beachten Sie: Die Symbole oder Befehle zum Bearbeiten dynamischer Tabellen **dürfen nicht** zum Bearbeiten statischer Tabellen verwendet werden.

Dynamische Tabellen enthalten Zeilen, die eine sich wiederholende Datenstruktur darstellen, d.h. jede Zeile weist dieselbe Datenstruktur auf (Dies ist bei statischen Tabellen nicht der Fall). Sie können daher Zeilen bearbeiten, also Zeilen anhängen, einfügen, nach oben oder nach unten verschieben und löschen. Diese Befehle stehen im Menü **Authentic** und als Symbole in der Symbolleiste (siehe Abb. unten) zur Verfügung.

Um diese Befehle zu verwenden, platzieren Sie den Cursor an eine beliebige Stelle in der gewünschten Zeile und wählen Sie den entsprechenden Befehl aus.

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.								

Zum Navigieren innerhalb der Zellen einer Tabelle verwenden Sie die Pfeiltasten der Tastatur (nach oben, nach unten, nach links, nach rechts). Um von einer Zelle zur nächsten zu springen, drücken Sie die Tabulatortaste. Wenn Sie in der letzten Zelle einer Zeile die Tabulatortaste drücken, wird eine neue Zeile erstellt.

XML-Tabellen

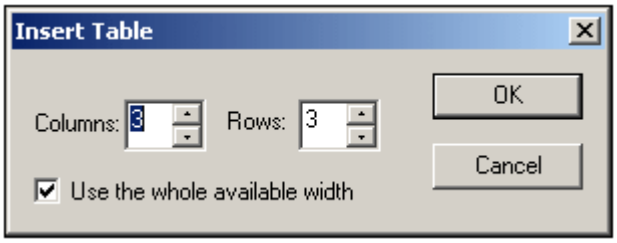
XML-Tabellen können von Ihnen, dem Benutzer der Authentic-Ansicht verwendet werden. Sie können damit Tabellen überall im XML-Dokument, wo dies zulässig ist, einfügen. Diese Funktion eignet sich zum Einfügen von Tabellendaten in Ihr Dokument. Diese Tabellen werden als Tabellen gedruckt, wenn Sie den Druckauftrag direkt von der Authentic-Ansicht aus geben. Wenn Sie Ausgabedokumente auch mit Hilfe von XSLT Stylesheets erzeugen, besprechen Sie die gewünschte Ausgabe mit dem Autor des StyleVision Power Stylesheet.

Beachten Sie bitte, dass XML-Tabellen nur an bestimmten Stellen, an denen dies zulässig ist, eingefügt werden können. Dies ist im Schema (DTD oder XML Schema) definiert. Wenn Sie Tabellen an anderen Stellen einfügen möchten, muss dies mit dem für das StyleVision Power Stylesheet Verantwortlichen besprochen werden.

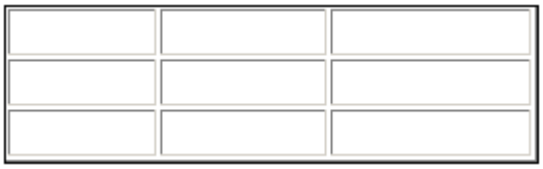
Arbeiten mit XML-Tabellen
Die Arbeit mit XML-Tabellen erfordert drei Schritte: Einfügen der Tabelle, Formatierung der Tabelle und Eingabe von Daten. Die Befehle für die Arbeit mit XML-Tabellen stehen in Form von Symbolen in der Symbolleiste zur Verfügung (siehe Symbole zur Bearbeitung von XML-Tabellen).

Einfügen von Tabellen
So fügen Sie eine XML-Tabelle ein:

1. Setzen Sie Ihren Cursor an die Stelle, an der die Tabelle eingefügt werden soll und klicken Sie auf das Symbol . (Beachten Sie dass XML-Tabellen nur an bestimmten im Schema definierten Stellen eingefügt werden können). Das Dialogfeld "Tabelle einfügen" (siehe unten) wird nun angezeigt.



2. Wählen Sie die Anzahl der Spalten und Zeilen aus und geben Sie an, ob die Tabelle sich über die gesamte Breite erstrecken soll. Daraufhin wird anhand der Angaben oben die folgende Tabelle erstellt.



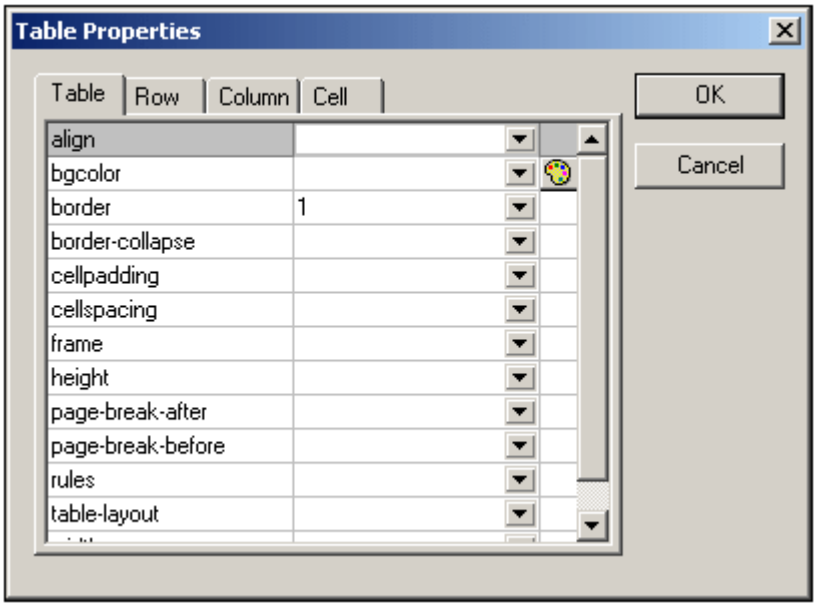
Sie können später Spalten hinzufügen und löschen, Zeilen erstellen und Spalten zusammenfügen. Erstellen Sie daher zuerst die grundlegende Struktur.

Bitte beachten Sie: Alle Änderungen an der Tabellenstruktur müssen mit Hilfe der Befehle des Tabellenmenüs vorgenommen werden. Sie können nicht durch Änderung von Attributwerten in der Attribut-Eingabehilfe gemacht werden.

Formatieren von Tabellen und Eingabe von Daten
So formatieren Sie Ihre Tabelle:

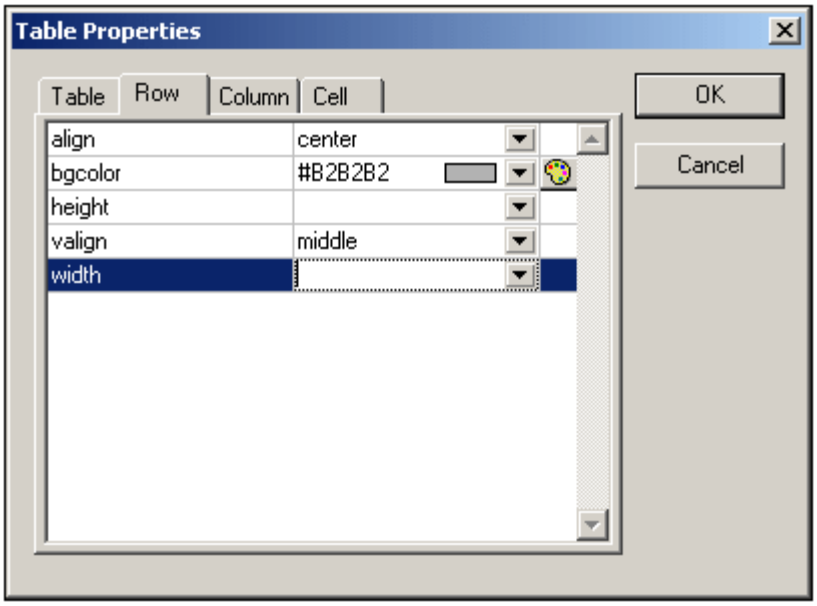
1. Platzieren Sie Ihren Cursor zum Formatieren der Tabelle an eine beliebige Stelle in der Tabelle und klicken Sie auf das Symbol (Tabelleneigenschaften). Daraufhin erscheint das Dialogfeld "Tabelleneigenschaften" (siehe unten), in dem Sie die Formatierung für die Tabelle, oder für einzelne Zeilen, Spalten oder Zellen definieren können.

© 2007 Altova GmbH



2. Setzen Sie die Eigenschaften "cellspacing" und "cellpadding" auf "0". Ihre Tabelle wird nun folgendermaßen aussehen:

3. Setzen Sie nun den Cursor in die erste Zeile, um diese zu formatieren und klicken Sie auf das Symbol (Tabelleneigenschaften). Klicken Sie auf das Register "Row".



Da es sich bei der ersten Zeile um die Kopfzeile handelt, definieren Sie eine Hintergrundfarbe, um diese Zeile von den anderen Zeilen zu unterscheiden. Beachten Sie dass die Zeileneigenschaften in der Abbildung oben definiert wurden. Geben Sie anschließend den Text für die Spaltenüberschriften ein. Ihre Tabelle sieht nun

folgendermaßen aus:

Name	Telephone	Email

4. Beachten Sie, dass die Ausrichtung wie angegeben mittig (centered) ist. Angenommen, Sie möchten nun die Spalte "Telephone" in die Spalten "Office" und "Home" unterteilen. Um dies zu tun, müssen Sie Zellen teilen. Setzen Sie die Cursor in die Zelle "Telephone" und klicken Sie auf das Symbol (Vertikal teilen). Ihre Tabelle sieht nun aus wie folgt:

Name	Telephone	Email

5. Setzen Sie nun den Cursor in die Zelle unterhalb der Zelle mit dem Eintrag "Telephone" und klicken Sie auf das Symbol (Horizontal teilen). Geben Sie dann die Spaltenüberschriften "Office" und "Home" ein. Ihre Tabelle sieht nun folgendermaßen aus:

Name	Telephone		Email
	Office	Home	

Nun müssen Sie die einzelnen Zellen in der Spalte "Telephone" vertikal teilen.

Mit Hilfe der Symbole zum Bearbeiten von Tabellen können Sie des Weiteren Spalten und Zeilen hinzufügen und löschen und Zellinhalte vertikal ausrichten. Die Symbole zum Bearbeiten von XML-Tabellen sind in der Benutzerreferenz im Abschnitt "XML-Tabellensymbole" näher beschrieben.

Navigieren innerhalb von Tabellenzellen
Um innerhalb der Zellen der XML-Tabelle navigieren zu können, verwenden Sie die Pfeiltasten der Tastatur.

Eingabe von Daten in eine Zelle
Um Daten in eine Zelle einzugeben, setzen Sie den Cursor in eine Zelle und geben Sie die Daten ein.

Formatieren von Text
Text in einer XML-Tabelle muss wie anderer Text im XML-Dokument mit Hilfe von XML-Elementen oder Attributen formatiert werden. Um ein Element hinzuzufügen, markieren Sie den Text und doppelklicken Sie auf das gewünschte Element in der Element-Eingabehilfe. Um einen Attributwert zu definieren, setzen Sie den Cursor in das Textfragment und geben Sie den erforderlichen Attributwert in der Attribut-Eingabehilfe ein. Nachdem Sie den Kopfzeilentext fett formatiert haben, sieht Ihre Tabelle folgendermaßen aus:

 © 2007 Altova GmbH

Name	Telephone		Email
	Office	Home	

Der Text in der Abbildung oben wurde durch Anklicken und Doppelklick auf das Element `strong` formatiert, für das eine globale Vorlage vorhanden ist, in der als Schriftbreite "Fett" definiert ist. Die Textformatierung wird sofort sichtbar.

Bitte beachten Sie: Damit die Textformatierung in der Authentic-Ansicht angezeigt wird, muss in StyleVision für das entsprechende Element eine globale Vorlage mit der erforderlichen Textformatierung erstellt worden sein.

Symbole zum Bearbeiten von XML-Tabellen

Die Befehle zum Bearbeiten von XML-Tabellen stehen in Form von Symbolen in der Symbolleiste zur Verfügung. Nachfolgend finden Sie eine Liste dieser Befehle. Beachten Sie, dass es zu diesen Symbolen keine Menübefehlentsprechungen gibt.

Ausführliche Informationen zur Verwendung von XML-Tabellen finden Sie unter XML-Tabellen.

Tabelle einfügen

Der Befehl "Tabelle einfügen" fügt eine **CALS / HTML-Tabelle** an der aktuellen Cursorposition ein.

Tabelle löschen

Der Befehl "Tabelle löschen" löscht die derzeit aktive Tabelle.

Zeile anhängen

Der Befehl "Zeile anhängen" hängt eine Zeile an das Ende der derzeit aktiven Tabelle an.

Spalte anhängen

Der Befehl "Spalte anhängen" hängt eine Spalte an das Ende der derzeit aktiven Tabelle an.

Zeile einfügen

Der Befehl "Zeile einfügen" fügt eine Zeile oberhalb der aktuellen Cursorposition in der derzeit aktiven Tabelle ein.

Spalte einfügen

Der Befehl "Spalte einfügen" fügt eine Spalte links von der derzeit aktiven Cursorposition in der derzeit aktiven Tabelle ein.

Links zusammenfügen

Der Befehl "Links zusammenfügen" fügt die aktive Zelle (an der Cursorposition) mit der Zelle links davon zusammen. Die Tags von beiden Zellen bleiben in der neuen Zelle erhalten, die Spaltenüberschriften bleiben unverändert.

Rechts zusammenfügen

Der Befehl "Rechts zusammenfügen" fügt die aktive Zelle (an der Cursorposition) mit der Zelle rechts davon zusammen. Die Tags von beiden Zellen bleiben in der neuen Zelle erhalten, die Spaltenüberschriften bleiben unverändert.

Unten zusammenfügen

Der Befehl "Unten zusammenfügen" fügt die aktive Zelle (an der Cursorposition) mit der Zelle unterhalb davon zusammen. Die Tags von beiden Zellen bleiben in der neuen Zelle erhalten, die Spaltenüberschriften bleiben unverändert.

Oben zusammenfügen

Der Befehl "Oben zusammenfügen" fügt die aktive Zelle (an der Cursorposition) mit der Zelle oberhalb davon zusammen. Die Tags von beiden Zellen bleiben in der neuen Zelle erhalten, die Spaltenüberschriften bleiben unverändert.

Horizontal teilen

Der Befehl "Horizontal teilen" erstellt eine neue Zelle rechts von der derzeit aktiven Zelle. Die Größe der beiden Zellen entspricht nun der Größe der Originalzelle

Vertikal teilen

Der Befehl "Vertikal teilen" erstellt eine neue Zelle unterhalb von der derzeit aktiven Zelle.

Bündig oben

Dieser Befehl richtet den Inhalt der Zellen bündig am oberen Zellenrand aus.

Vertikal zentrieren

Dieser Befehl zentriert den Inhalt der Zelle.

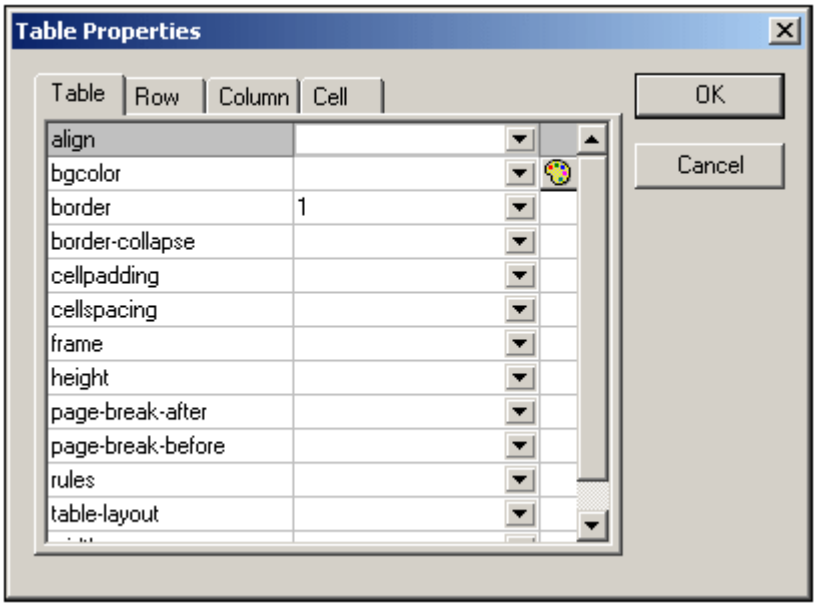
Bündig unten

Dieser Befehl richtet den Inhalt der Zellen bündig am unteren Zellenrand aus.

Tabelleneigenschaften

Der Befehl "Tabelleneigenschaften" öffnet das Dialogfeld "Tabelleneigenschaften". Dieses Symbol wird nur bei HTML-Tabellen aktiv, bei CALS-Tabellen ist es deaktiviert.

© 2007 Altova GmbH



6.2.2 Bearbeiten einer DB

In der Authentic-Ansicht können Sie Datenbank (DB)-Tabellen bearbeiten und die Daten in einer DB speichern. Dieser Abschnitt enthält eine ausführliche Beschreibung der Funktionen, die Ihnen zum Bearbeiten einer DB-Tabelle zur Verfügung stehen. Beachten Sie bitte die folgenden Punkte:

- Die Anzahl der Datensätze in einer DB-Tabelle, die in der Authentic-Ansicht angezeigt werden können, wurde unter Umständen vom Autor des StyleVision Power Stylesheet absichtlich eingeschränkt, um das Design kompakter zu gestalten. In diesem Fall wird anfangs nur diese beschränkte Anzahl an Datensätzen in der Authentic-Ansicht geladen. Sie können die restlichen Datensätze der DB-Tabelle mit Hilfe der Symbole zur Navigation in DB-Tabellenzeilen (*siehe* Navigieren in einer DB-Tabelle) laden und anzeigen.
- Sie können eine Datenbankabfrage durchführen, um bestimmte Datensätze anzuzeigen.
- Sie können Datensätze hinzufügen, ändern und löschen und Ihre Änderungen in der DB speichern. Siehe Ändern einer DB-Tabelle.

So öffnen Sie ein DB-basiertes StyleVision Power Stylesheet in der Authentic Ansicht:

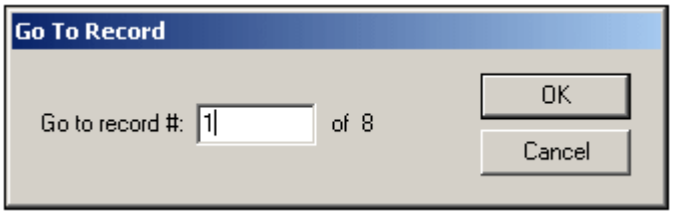
- Klicken Sie auf **Authentic | Datenbank bearbeiten**, und suchen Sie nach dem gewünschten StyleVision Power Stylesheet.

Navigieren in einer DB-Tabelle

Die Befehle zum Navigieren in den Zeilen einer DB-Tabelle stehen in Form von Schaltflächen im Authentic-Ansichtsdokument zur Verfügung. Normalerweise gibt es zu jeder DB-Tabelle einen Navigationsbereich mit vier oder fünf Schaltflächen.

Die Pfeilsymbole von links nach rechts sind: Gehe zum ersten Datensatz in der DB; Gehe zum vorhergehenden Datensatz; Dialogfeld "Gehe zu Datensatz #" öffnen (*siehe Abbildung unten*);

Gehe zum nächsten Datensatz; und Gehe zum letzten Datensatz.



Um durch eine DB-Tabelle zu navigieren, klicken Sie auf die gewünschte Schaltfläche.

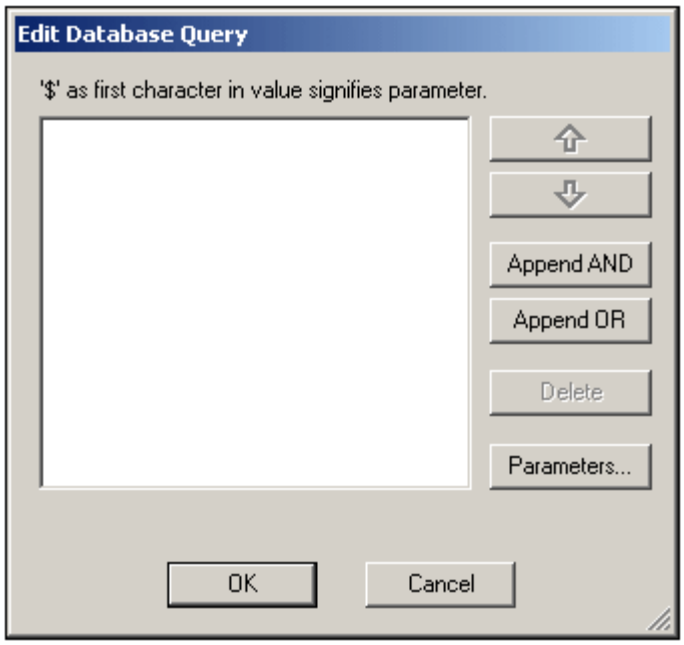
DB-Abfragen

Mit Hilfe einer DB-Abfrage können Sie die Datensätze einer in der Authentic-Ansicht angezeigten Tabelle abfragen. Eine Abfrage wird für eine einzelne Tabelle erstellt. Es kann für jede Tabelle nur eine Abfrage erstellt werden. Sie können während der Bearbeitung eines Dokuments jederzeit eine Abfrage erstellen. Wenn das in der Authentic-Ansicht angezeigte Dokument zu dem Zeitpunkt, zu dem Sie die Abfrage absenden, nicht gespeicherte Änderungen enthält, werden Sie gefragt, ob **alle** Änderungen im Dokument gespeichert oder verworfen werden sollen. Beachten Sie, dass auch an anderen Tabellen vorgenommene Änderungen gespeichert/verworfen werden. Nach Absenden der Abfrage wird die Tabelle mit den aktuellen Abfragekriterien erneut geladen.

Bitte beachten Sie: Falls eine Meldung angezeigt wird, dass zu viele Tabellen geöffnet sind, können Sie die Anzahl der offenen Tabellen mit Hilfe einer Abfrage durch Filtern reduzieren.

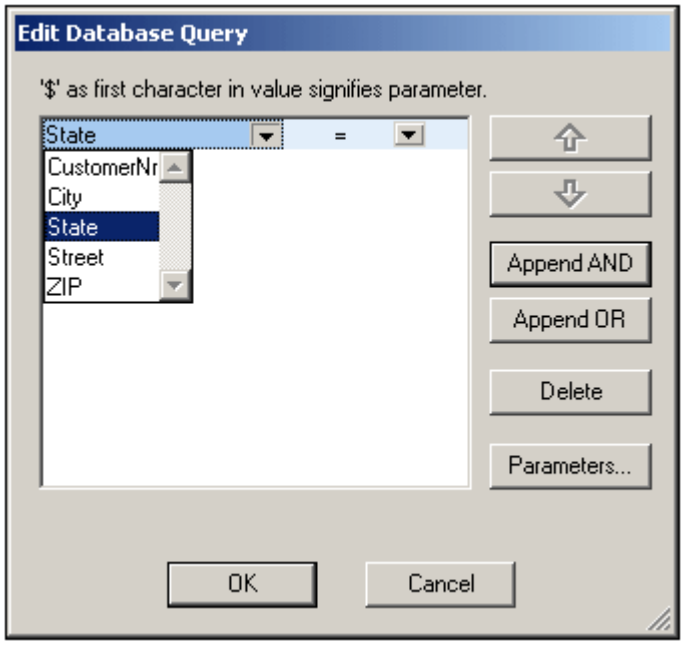
So erstellen Sie eine Abfrage und senden diese ab:

1. Klicken Sie auf die Schaltfläche "Abfrage" für die entsprechende Tabelle, um das Dialogfeld "Datenbankabfrage bearbeiten" aufzurufen (*siehe Abbildung unten).* Diese Schaltfläche erscheint normalerweise am oberen Rand einer DB-Tabelle oder unterhalb davon. Wenn keine Abfrage-Schaltfläche vorhanden ist, hat der Autor des StyleVision Power Stylesheet die DB-Abfragefunktion für diese Tabelle nicht aktiviert.



 © 2007 Altova GmbH

2. Klicken Sie auf die Schaltfläche **AND anhängen** oder **OR anhängen**. Daraufhin wird ein leeres Abfragekriterium (siehe unten) angehängt.



4. Geben Sie den Ausdruck für das Kriterium ein. Ein Ausdruck besteht aus: (i) einem Feldnamen (der in der dazugehörigen Dropdown-Liste zur Verfügung steht); (ii) einem Operator (der in der zugehörigen Dropdown-Liste zur Verfügung steht); und (iii) einem Wert (direkt einzugeben). Nähere Informationen zum Erstellen von Ausdrücken finden Sie im Abschnitt Ausdrücke in Kriterien weiter unten.
5. Wenn Sie ein weiteres Kriterium hinzufügen möchten, klicken Sie - je nachdem, welchen logischen Operator (AND oder OR) Sie verwenden wollen, um Ihre Kriterien zu verknüpfen - auf die Schaltfläche **AND anhängen** bzw. **OR anhängen.** Fügen Sie nun das neue Kriterium hinzu. Nähere Informationen zu logischen Operatoren finden Sie im Abschnitt Neuordnen von Kriterien in DB-Abfragen.

Ausdrücke in Kriterien

Ausdrücke in DB-Abfragekriterien bestehen aus einem Feldnamen, einem Operator und einem Wert. Als **Feldnamen** stehen die Child-Elemente der obersten Ebene der Datentabelle zur Verfügung; die Namen dieser Felder werden in der Dropdown-Liste aufgelistet (*siehe Abbildung oben)*. Unterhalb sehen Sie eine Liste der verfügbaren **Operatoren**:

```
=             Ist gleich
<>            Ist nicht gleich
<             Kleiner als
<=            Kleiner oder gleich
>             Größer als
>=            Größer oder gleich
LIKE          Phonetisch ähnlich
NOT LIKE      Phonetisch nicht ähnlich
IS NULL       Ist leer
NOT NULL      Ist nicht leer
```

Wenn IS NULL oder NOT NULL ausgewählt ist, ist das Wertefeld deaktiviert. **Werte** müssen ohne Anführungszeichen (oder andere Begrenzungszeichen) eingegeben werden. Außerdem müssen Werte dieselbe Formatierung wie die des entsprechenden DB-Felds haben, sonst ist das Ergebnis des Ausdrucks `FALSE`. Wenn z.B. ein Kriterium für ein Feld des `date`-Datentyps

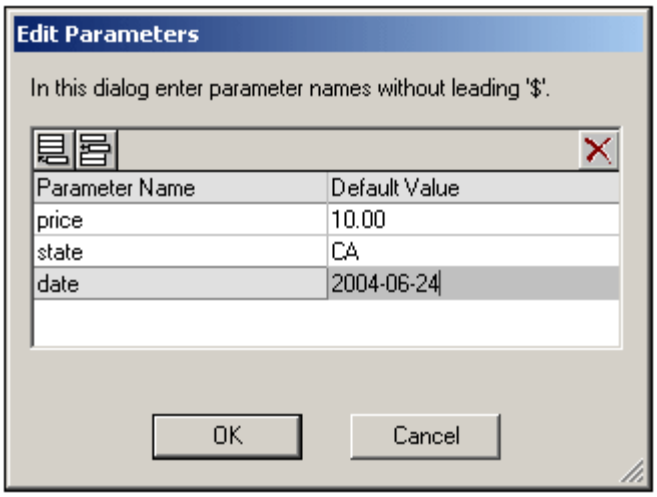
in einer MS Access DB einen Ausdruck `StartDate=25/05/2004` enthält, ist das Ergebnis der Abfrage `FALSE`, da der `date`-Datentyp in einer MS Access DB das Format `JJJJ-MM-TT` hat.

Verwendung von Parametern mit DB-Abfragen
Sie können beim Erstellen von Abfragen den Namen eines **Parameters** als Wert eines Ausdrucks eingeben. Parameter sind Variablen, die anstelle von Literal-Werten in Abfragen verwendet werden können. Zuerst müssen Sie den Parameter und seinen Wert deklarieren und ihn anschließend in einem Ausdruck verwenden. Der Wert des Parameters wird als der Wert dieses Ausdrucks verwendet. Bei den Parametern, die Sie im Dialogfeld "Parameter bearbeiten" hinzufügen, handelt es sich um solche, die bereits für das Stylesheet deklariert wurden. In diesem Fall setzt der neue Wert den Wert im Stylesheet außer Kraft.

Parameter sind nützlich, wenn Sie einen einzelnen Wert in mehreren Ausdrücken verwenden wollen.

Deklarieren von Parametern über das Dialogfeld "Datenbankabfrage bearbeiten"
So deklarieren Sie Parameter:

1. Klicken Sie im Dialogfeld **Datenbankabfrage bearbeiten** auf die Schaltfläche **Parameter...**, um das unten gezeigte Dialogfeld **Parameter bearbeiten** aufzurufen.



2. Klicken Sie auf Anhängen oder Einfügen
2. Geben Sie den Namen und Wert des Parameters in die entsprechenden Felder ein.

Bitte beachten Sie: Das Dialogfeld "Parameter bearbeiten" enthält **alle** Parameter, die für das Stylesheet definiert wurden. Es ist zwar ein Fehler, im StyleVision Power Stylesheet einen nicht deklarierten Parameter zu verwenden, doch ist es kein Fehler, einen Parameter zu deklarieren und diesen nicht zu verwenden.

Verwendung von Parametern in Abfragen
Um den Namen eines Parameters als Wert eines Ausdrucks zu verwenden:

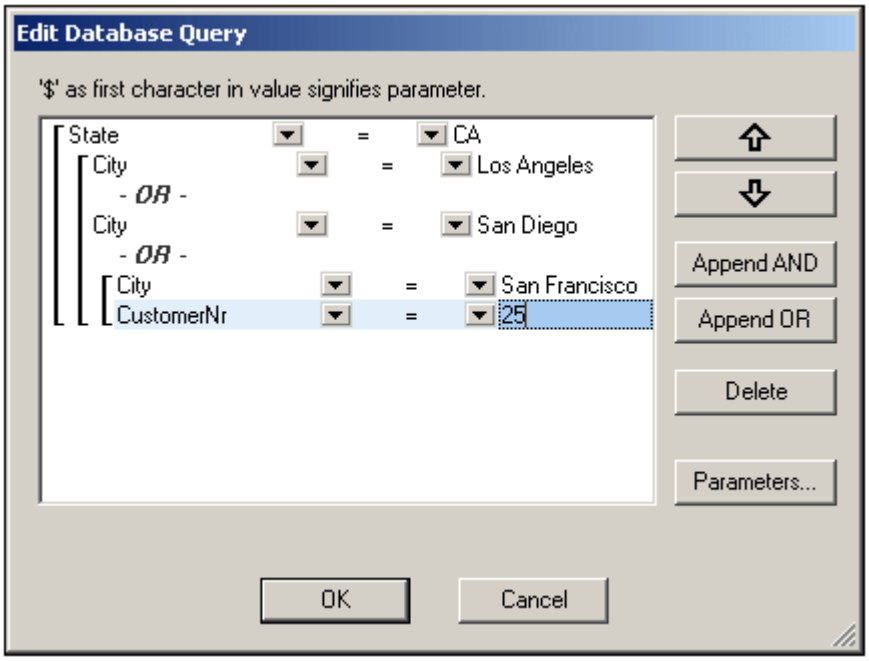
- Geben Sie das `$`-Zeichen in das Werteingabefeld ein, gefolgt vom Namen des Parameters (ohne Leerzeichen).

Bitte beachten Sie: Wenn der Parameter bereits deklariert wurde, wird der Eintrag grün angezeigt. Wenn der Parameter noch nicht deklariert wurde, erscheint der Eintrag rot und der Parameter muss deklariert werden.

 © 2007 Altova GmbH

Neuordnen von Kriterien in DB-Abfragen

- Die logische Struktur der DB-Abfrage und die Beziehung zwischen zwei Kriterien oder Kriteriengruppen wird grafisch angezeigt. Jede Ebene der logischen Struktur wird durch eine eckige Klammer angezeigt. Zwei Kriterien oder Kriteriengruppen nebeneinander bedeuten, dass der AND-Operator verwendet wurde, während bei Trennung von zwei Kriterien durch das Wort OR der OR-Operator verwendet wurde. Die Kriterien dienen auch dazu, dem Benutzer einen klaren Überblick über die logische Struktur der DB-Abfrage zu verschaffen.



Die in der obigen Abbildung gezeigte DB-Abfrage würde in Textform folgendermaßen aussehen:

```
State=CA AND (City=Los Angeles OR City=San Diego OR (City=San
Francisco AND CustomerNr=25))
```

Sie können die DB-Abfrage umstrukturieren, indem Sie ein Kriterium oder eine Gruppe von Kriterien im Vergleich zu den anderen Kriterien in der DB-Abfrage nach oben oder unten verschieben. Um ein Kriterium oder eine Gruppe von Kriterien zu verschieben, gehen Sie vor wie folgt:

1. Wählen Sie das Kriterium durch Anklicken aus oder wählen Sie eine ganze Ebene aus, indem Sie auf die Klammer für diese Ebene klicken.
2. Klicken Sie im Dialogfeld auf den Nach oben- oder Nach unten-Pfeil.

Beachten Sie die folgenden Punkte:

- Wenn sich das in Richtung der Verschiebung benachbarte Kriterium auf derselben Ebene befindet, werden die beiden Kriterien vertauscht.
- Eine Gruppe von Kriterien (d.h. ein Kriterium mit einer Klammer) ändert seine Position innerhalb derselben Ebene; die Ebene ändert sich nicht.
- Ein einzelnes Kriterium ändert seine Position innerhalb derselben Ebene. Wenn sich das benachbarte Kriterium weiter außen/innen (d.h. nicht auf derselben Ebene) befindet, wird das ausgewählte Kriterium **immer um eine Ebene** nach außen/innen verschoben.

Um ein Kriterium in einer DB-Abfrage zu löschen, wählen Sie es aus und klicken Sie auf **Löschen**.

Ändern einer DB-Abfrage
So ändern Sie eine DB-Abfrage:

1. Klicken Sie auf die Schaltfläche "Abfrage" , um das Dialogfeld "Datenbankabfrage bearbeiten" aufzurufen. Sie können nun die Ausdrücke in jedem der aufgelisteten Kriterien ändern, neue Kriterien hinzufügen oder Kriterien aus der DB-Abfrage löschen.
2. Klicken Sie auf OK. Die Daten aus der DB werden automatisch in StyleVision neu geladen, um die Änderungen an der DB-Abfrage anzuzeigen.

Ändern einer DB-Tabelle

Hinzufügen eines Datensatzes
So fügen Sie einen Datensatz hinzu:

1. Setzen Sie den Cursor in die gewünschte Zeile der DB-Tabelle und klicken Sie auf das Symbol , (um eine Zeile anzuhängen) oder (um eine Zeile einzufügen). Daraufhin wird ein neuer Datensatz in der temporären XML-Datei erstellt.
2. Klicken Sie auf den Befehl **Datei | Speichern...**, um den neuen Datensatz zur Datenbank hinzuzufügen. In der Authentic-Ansicht wird in der Datenbank-Tabelle eine neue Zeile für den neuen Datensatz angehängt. Der `AltovaRowStatus` für diesen Datensatz wird auf `A` (für Added = Hinzugefügt) gesetzt.

Wenn Sie Daten für den neuen Datensatz eingeben, werden diese fett und unterstrichen angezeigt. Auf diese Art können Sie neu hinzugefügte Datensätze von vorhandenen unterscheiden - wenn vorhandene Datensätze nicht mit diesen Texteigenschaften formatiert wurden. Datentypfehler werden rot gekennzeichnet.

Der neue Datensatz wird zur DB hinzugefügt, wenn Sie **Datei | Speichern...** auswählen. Nachdem ein neuer Datensatz zur Datenbank hinzugefügt wurde, wird das `AltovaRowStatus` -Feld initialisiert (gekennzeichnet durch `---`) und der Datensatz wird in der Authentic-Ansicht als regulärer Datensatz angezeigt.

Ändern eines Datensatzes
Um einen Datensatz zu ändern, setzen Sie den Cursor an die gewünschte Stelle in der DB-Tabelle und nehmen Sie die gewünschten Änderungen vor. Wenn die Anzahl der Datensätze beschränkt ist, müssen Sie eventuell (mit Hilfe der oben beschriebenen Navigationssymbole) zum gewünschten Datensatz navigieren.

Wenn Sie einen Datensatz bearbeiten, werden die Einträge in allen Feldern des Datensatzes unterstrichen und der `AltovaRowStatus` aller primären Instanzen dieses Datensatzes wird auf `U` (für Updated = Aktualisiert) gesetzt. Für alle sekundären Instanzen dieses Datensatzes wird der `AltovaRowStatus` auf `u` (klein geschrieben) gesetzt. Die primären und sekundären Instanzen eines Datensatzes werden durch die Struktur der DB definiert - und ebenfalls von dem davon generierten XML-Schema. Wenn z.B. eine Adresstabelle in einer Kundentabelle enthalten ist, kann die Adresstabelle im Design-Dokument in zwei Instantiierungstypen vorkommen: als Adresstabelle selbst und innerhalb von Instantiierungen der Kundentabelle. Derjenige der beiden Typen, der geändert wird, ist der primär geänderte Typ. Andere Typen - es kann mehr als einen anderen Typ geben - sind sekundäre Typen. Datentypfehler werden rot markiert.

Sie speichern die Änderungen in der DB durch Klicken auf **Datei | Speichern...**. Nachdem ein

geänderter Datensatz in der DB gespeichert wurde, wird das `AltovaRowStatus` Feld initialisiert (gekennzeichnet durch `---`) und der Datensatz wird in der Authentic-Ansicht als regulärer Datensatz angezeigt.

Bitte beachten Sie:

- Selbst wenn nur ein einziges Feld eines Datensatzes in der Authentic-Ansicht geändert wird, wird der gesamte Datensatz beim Speichern der Daten in der Datenbank aktualisiert.
- Der Datumswert `0001-01-01` ist bei einigen Datenbanken als `NULL`-Wert definiert und könnte daher eine Fehlermeldung verursachen.

Löschen eines Datensatzes
So löschen Sie einen Datensatz:

1. Setzen Sie den Cursor in die Zeile für den entsprechenden Datensatz und klicken Sie auf das Symbol . Der zu löschende Datensatz erscheint nun durchgestrichen. Der `AltovaRowStatus` erhält folgenden Wert: Primäre Instanzen des Datensatzes werden auf `D gesetzt;` sekundäre Instanzen auf `d`; indirekt gelöschte Datensätze werden mit `X` gekennzeichnet. Indirekt gelöschte Datensätze sind Felder im gelöschten Datensatz, die in einer separaten Tabelle enthalten sind. So könnte z.B. in einer Kundentabelle eine Adresstabelle enthalten sein. Wenn ein Kundendatensatz gelöscht würde, würde indirekt auch der entsprechende Adressdatensatz gelöscht. Wenn in der Kundentabelle ein Adressdatensatz gelöscht würde, würde der Adressdatensatz in der Kundentabelle primär gelöscht, jedoch würde derselbe Datensatz in einer unabhängigen Adresstabelle sekundär gelöscht, wenn diese instantiiert würde.
2. Sie speichern die Änderungen in der DB durch Klicken auf **Datei | Speichern...**.

Bitte beachten Sie: Wenn Sie Daten in der DB speichern, wird der Rückgängig-Befehl zurückgesetzt. Sie können Operationen, die vor dem Speichern durchgeführt wurden, nicht mehr rückgängig machen.

6.2.3 Arbeiten mit Datumsangaben

Es gibt in der Authentic-Ansicht zwei Methoden, um Datumsangaben zu bearbeiten:

- Die Eingabe oder Änderung des Datums erfolgt über die Datumswahl.
- Das Datum wird durch Eingabe des Werts eingegeben bzw. geändert.

Welche Methode der Benutzer der Authentic-Ansicht verwendet, ist im StyleVision Power Stylesheet definiert. Beide Methoden sind in den beiden Unterkapiteln dieses Abschnitts beschrieben.

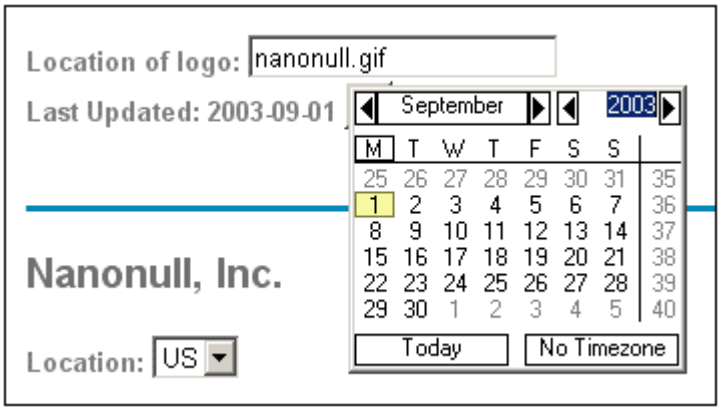
Anmerkung zu Datumsformaten
Zur Speicherung des Datums im XML-Dokument stehen mehrere Datums-Datentypen zur Verfügung. Bei jedem dieser Datums-Datentypen muss das Datum in einem bestimmten lexikalischen Format gespeichert werden, damit das XML-Dokument gültig ist. So muss das Datum z.B. beim Datentyp `xs:date` das lexikalische Format `JJJJ-MM-TT` aufweisen. Wenn das Datum in einem `xs:date` Node anders als in diesem Format eingegeben wurde, ist das XML-Dokument ungültig.

Um sicherzustellen, dass das Datum im richtigen Format eingegeben wird, kann der Designer des StyleVision Power Stylesheet die grafische Datumswahl in das Design integrieren. Dadurch wird sichergestellt, dass das mit der Datumswahl ausgewählte Datum im richtigen lexikalischen Format eingegeben wird. Falls keine Datumswahl vorhanden ist, sollte in der Authentic-Ansicht sichergestellt werden, dass das Datum im korrekten Format eingegeben wird. Bei der Validierung des XML-Dokuments erhalten Sie nützliche Tipps über das erforderliche lexikalische Format.

Datumswahl

Die Datumswahl ist ein grafischer Kalender zur Eingabe eines Datums in einem Standardformat in ein XML-Dokument. Es ist für die Verarbeitung des Dokuments wichtig, dass ein Standardformat verwendet wird. Das Datumswahlsymbol erscheint neben dem Datumsfeld, das dadurch geändert wird (siehe Abbildung unten).

Um die Datumswahl (siehe unten) anzuzeigen, klicken Sie auf das Datumswahl-Symbol.



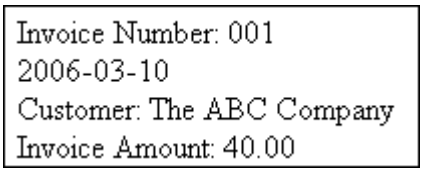
Um ein Datum auszuwählen, klicken Sie auf das gewünschte Datum, den gewünschten Monat bzw. das gewünschte Jahr. Das Datum wird in das XML-Dokument eingegeben und das angezeigte Datum wird entsprechend geändert. Falls nötig können Sie auch eine Zeitzone angeben.

Texteintrag

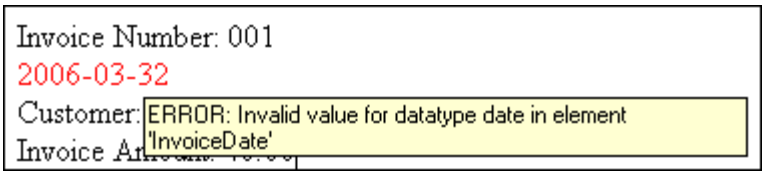
In Datumsfeldern, die keine Datumswahl haben (*siehe Screenshot*) können Sie das Datum direkt durch Eingabe des neuen Werts bearbeiten.

Bitte beachten Sie: Beim Bearbeiten des Datums darf das Format nicht geändert werden.

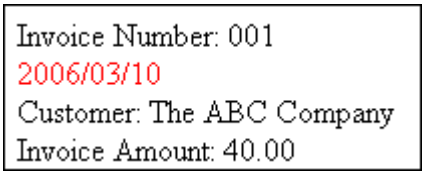
© 2007 Altova GmbH



Wenn Sie ein Datum ändern und es sich daraufhin außerhalb des gültigen Bereichs für Datumsangaben befindet, wird das Datum rot angezeigt, um den Fehler anzuzeigen. Wenn Sie den Mauszeiger über das ungültige Datum platzieren, erscheint eine Fehlermeldung (*siehe Screenshot*).



Wenn Sie versuchen, das Datumsformat zu ändern, wird das Datum rot angezeigt (*siehe Screenshot*).



6.2.4 Definieren von Entities

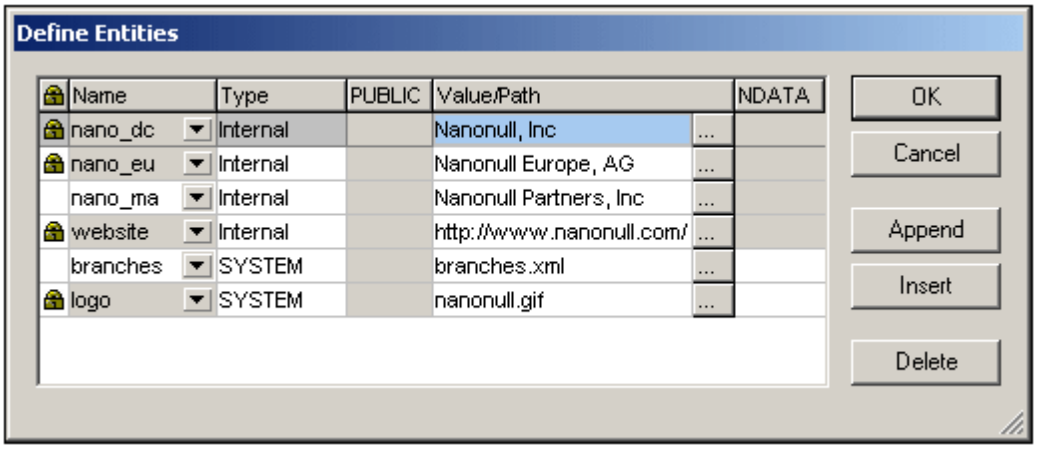
Unabhängig davon, ob Ihr Dokument auf einer DTD oder einem XML-Schema basiert, können Sie für die Verwendung in der Authentic-Ansicht Entities definieren. Diese Entities werden anschließend in der Entities-Eingabehilfe und im Kontextmenü-Untermenü von **Entity einfügen** angezeigt. Wenn Sie in der Entities-Eingabehilfe auf eine Entity doppelklicken, wird diese an der Cursorposition eingefügt.

Eine Entity eignet sich v.a. dann, wenn Sie einen Textstring, ein XML-Fragment oder einige andere externe Ressourcen an mehreren Stellen in Ihrem Dokument verwenden möchten. Die Entity, bei der es sich im Grunde um eine Kurzbezeichnung für die benötigten Daten handelt, wird im Dialogfeld "Entities definieren" definiert. Nachdem Sie die Entity definiert haben, können Sie diese an mehreren Stellen in Ihrem Dokument verwenden und dadurch Zeit sparen und effizienter arbeiten.

Sie können in Ihrem Dokument zwei grundlegende Arten von Entities verwenden: **geparste Entities**, wobei es sich um XML-Daten handelt (entweder einen Textstring oder ein Fragment aus einem XML-Dokument) oder **ungeparste Entities**, wobei es sich um nicht-XML-Daten handelt wie z.B. eine Binärdatei (normalerweise eine Grafik, Ton oder ein Multimedia-Objekt). Jede Entity hat einen Namen und einen Wert. Im Fall von geparsten Entities ist die Entity ein Platzhalter für die XML-Daten. Der Wert der Entity sind entweder die XML-Daten selbst oder eine URI, die auf eine `.xml`-Datei verweist, die die XML-Daten enthält. Bei ungeparsten Entities ist der Wert der Entity eine URI, die auf eine nicht-XML-Datei verweist.

So definieren Sie eine Entity:

1. Klicken Sie auf **Authentic | XML-Entities definieren...**. Daraufhin wird das Dialogfeld "Entities definieren" geöffnet. (*Abbildung unten*)



2. Geben Sie den Namen Ihrer Entity im Feld "Name" ein. Dies ist der Name, der in der Entities-Eingabehilfe angezeigt werden wird.
3. Wählen Sie den Typ der Entity aus der Dropdown-Liste im Feld "Type" aus. Es stehen die folgenden Typen zur Auswahl. Eine **interne** (internal) Entity ist eine Entity, für die der zu verwendende Text im XML-Dokument selbst gespeichert ist. Wenn Sie **PUBLIC** oder **SYSTEM** auswählen, wird angegeben, dass sich die Ressource außerhalb der XML-Datei befindet und mittels eines "Public Identifier" oder eines "System Identifier" gefunden wird. Ein System Identifier ist eine URI, die den Pfad der Ressource definiert. Ein Public Identifier ist ein pfadunabhängiger Identifier, der es manchen Prozessoren ermöglicht, die Ressource zu identifizieren. Wenn Sie sowohl einen Public als auch einen System Identifier angeben, wird der Public Identifier zum System Identifier aufgelöst und es wird der System Identifier verwendet.
4. Wenn Sie als Type PUBLIC ausgewählt haben, geben Sie den Public Identifier der Ressource im Feld PUBLIC ein. Wenn Sie Internal oder SYSTEM als Type ausgewählt haben, ist das Feld PUBLIC deaktiviert.
5. Im Feld "Value/Path" haben Sie folgende Eingabemöglichkeiten:
   - Wenn die Entity "Internal" ist, geben Sie den Textstring ein, der als Wert der Entity verwendet werden soll. Setzen Sie den Textstring nicht in Anführungszeichen. Alle Anführungszeichen werden als Teil des Textstrings behandelt.
   - Wenn der Entity-Typ SYSTEM ist, geben Sie die URI der Ressource ein oder wählen Sie mit Hilfe der Durchsuchen-Schaltfläche eine Ressource auf Ihrem lokalen Netzwerk aus. Wenn die Ressource geparste Daten enthält, muss es sich um eine XML-Datei handeln (d.h. die Datei muss die Erweiterung . `xml` haben). Als Alternative dazu kann die Ressource eine Binärdatei sein, z.B. eine GIF-Datei.
   - Wenn der Entity-Typ PUBLIC ist, müssen Sie zusätzlich einen System Identifier in dieses Feld eingeben.
6. Der Eintrag NDATA sagt dem Prozessor, dass diese Entity nicht geparst, sondern an den entsprechenden Prozessor gesendet werden soll. Das Feld "NDATA" sollte daher nur mit ungeparsten Entities verwendet werden.

Dialogfeldfunktionen

Im Dialogfeld "Entities definieren" können Sie Folgendes tun:

- Entities anhängen
- Entities einfügen
- Entities löschen
- Entities spaltenweise alphabetisch sortieren, indem Sie auf die entsprechende Spaltenüberschrift klicken; bei einem einmaligen Klick wird aufsteigend sortiert, bei zweimaligem Klick absteigend
- die Größe des Dialogfelds und die Spaltenbreite anpassen.
- Sperren. Sobald eine Entity im XML-Dokument verwendet wird, ist sie gesperrt und

 © 2007 Altova GmbH

kann im Dialogfeld "Entities definieren" nicht bearbeitet werden. Gesperrte Entities werden durch das Schlosssymbol in der ersten Spalte gekennzeichnet. Durch das Sperren einer Entity wird sichergestellt, dass das Dokument in Bezug auf Entities gültig ist. (Das Dokument wäre ungültig, wenn eine Entity referenziert aber nicht definiert ist.)
- Doppelt vorhandene Entities werden markiert.

Einschränkungen

- Eine Entity innerhalb einer anderen Entity wird nicht aufgelöst - weder im Dialogfeld, noch in der Authentic-Ansicht, noch in der XSLT-Ausgabe, und das kaufmännische Und-Zeichen einer derartigen Entity wird in seiner Escape-Form, also `&` angezeigt.
- Externe Entities werden in der Authentic-Ansicht nicht aufgelöst, es sei denn es handelt sich bei der Entity um eine Bilddatei und sie wurde als der Wert eines Attributs vom Typ `ENTITY` oder `ENTITIES` eingegeben. Derartige Entities werden aufgelöst, wenn das Dokument mit einem anhand des SPS generierten XSLT verarbeitet wird.

6.2.5 Grafiken in der Authentic-Ansicht

Die Authentic-Ansicht basiert auf dem Internet Explorer und kann die meisten der Grafikformate, die von Ihrer Internet Explorer-Version unterstützt werden, anzeigen. Unterstützt werden die folgenden gebräuchlichen Grafikformate:

- GIF
- JPG
- PNG
- BMP
- WMF (Microsoft Windows Metafile)
- EMF (Enhanced Metafile)
- SVG (nur für PDF-Ausgabe)

Relative Pfade
Relative Pfade werden relativ zur SPS-Datei aufgelöst.

6.2.6 Verwendung von Tasten in der Authentic-Ansicht

Eingabetaste
In der Authentic-Ansicht wird die **Eingabetaste** an bestimmten Cursorpositionen verwendet, um zusätzliche Elemente anzuhängen. Wenn das Kapitel eines Buchs z.B. (laut Schemadefinition) mehrere Absätze enthalten kann, wird bei Drücken der **Eingabetaste** innerhalb des Texts des Absatzes ein neuer Absatz direkt hinter dem aktuellen Absatz angehängt. Wenn ein Kapitel einen Titel und mehrere Kapitel enthalten kann, wird bei Drücken der **Eingabetaste** innerhalb des Kapitels aber außerhalb eines Absatzelements (auch innerhalb des Titel-Elements) ein neues Kapitel hinter dem aktuellen Kapitel angehängt (vorausgesetzt es sind laut Schemadefinition mehrere Kapitel zulässig).

Bitte beachten Sie: Bei Drücken der **Eingabetaste** wird keine Zeilenschaltung, kein Zeilenvorschub eingefügt, d.h. es wird keine neue Zeile angelegt. Dies ist auch dann der Fall, wenn sich der Cursor innerhalb eines Textnode wie z.B. eines Absatzes befindet.

7 Browser-Ansicht

Die Browser-Ansicht wird normalerweise zur Ansicht folgender Dateien verwendet:

- XML-Dateien, die mit einer XSLT-Datei verknüpft wurden. Wenn Sie in die Browser-Ansicht wechseln, wird die XML-Datei mit Hilfe eines XSLT Stylesheet on-the-fly transformiert. Das Ergebnis wird direkt in der Browser-Ansicht angezeigt.
- HTML-Dateien, die direkt im HTML-Format erstellt wurden oder mittels einer XSLT-Transformation einer XML-Datei erzeugt wurden.

Für die Browser-Ansicht wird Microsoft Internet Explorer 5.0 oder höher benötigt. Zur Anzeige von XML-Dateien, die mit Hilfe eines XSLT Stylesheet transformiert wurden, sollten Sie unbedingt Internet Explorer 6.0 oder höher verwenden, da diese Versionen MSXML 3.0 verwenden, einen XML-Parser, die den XSLT 1.0 Standard vollständig unterstützt. Eventuell sollten Sie auch MSXML 4.0 installieren.

Nähere Informationen dazu erhalten Sie in unserem Download Center. (Beachten Sie, dass die Unterstützung von XSL in IE 5 nicht zu 100 % kompatibel mit der offiziellen XSLT Recommendation ist. Wenn Sie daher in der Browser-Ansicht Probleme mit IE 5 haben, sollten Sie auf IE 6 upgraden.)

Um XML-und HTML-Dateien in der Browser-Ansicht zu sehen, klicken Sie auf das Register "Browser".

Funktionen der Browser-Ansicht

- Die Browser-Ansicht lässt sich in einem separaten Fenster öffnen. Wechseln Sie dazu in die Browser-Ansicht und wählen Sie den Menübefehl **Browser | Neues Fenster**. Auf diese Art können Sie Fenster nebeneinander anordnen und so die Browser-Ansicht Seite an Seite mit der Bearbeitungsansicht anzeigen. Alle Änderungen, die Sie in der Bearbeitungsansicht vornehmen, sind sofort in der Browser-Ansicht sichtbar: Drücken Sie in der Bearbeitungsansicht einfach die Taste **F5** oder machen Sie die Browser-Ansicht zum aktiven Fenster (indem Sie darauf klicken).
- Die Browser-Ansicht unterstützt die Suchfunktion. Wählen sie dazu in der Browser-Ansicht die Menübefehl **Bearbeiten | Suchen**, um Textstrings zu suchen.
- Die Browser-Ansicht unterstützt gebräuchliche Browserbefehle: Zurück, Vorwärts, Abbrechen, Aktualisieren, Schriftgröße und Drucken.

© 2007 Altova GmbH

8 Archivansicht

Bei einer ZIP-Datei kann es sich um eine von zwei Arten handeln:

- ein Archiv einer oder mehrerer Dateien, die komprimiert wurden (z.B. eine WinZip-Datei)
- einen Wrapper für mehrere Dateien in einer bestimmten Struktur (z.B. eine Microsoft Office Open XML-Datei)

In der Archivansicht bietet XMLSpy native Unterstützung für ZIP-Dateien. ZIP-Dateien können in der Archivansicht erstellt oder geöffnet werden und in dieser Ansicht auch in ihrer Struktur geändert werden. Einzelne Dateien im Archiv können von der Archivansicht aus geöffnet, in einer der Editieransichten von XMLSpy bearbeitet und anschließend direkt im ZIP-Archiv gespeichert werden.

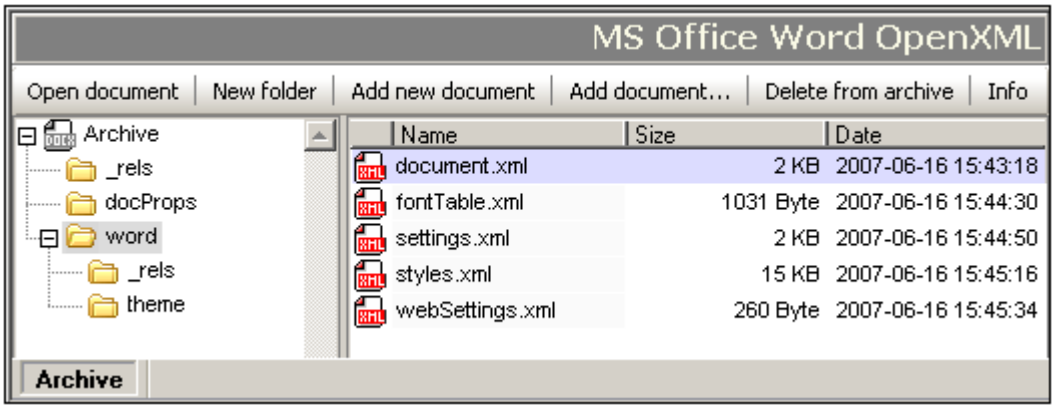
Zusätzlich dazu unterstützt XMLSpy die Bearbeitung von MS Word Open XML- und MS Excel Open XML-Dokumenten (Eingabehilfen, Autokomplettierung und Validierung).

Diese Beschreibung der Archivansicht ist in die folgenden Abschnitte gegliedert:

- eine Beschreibung der Benutzeroberfläche der Archivansicht und ihrer Teile
- eine Beschreibung der einzelnen Operationen, die in XMLSpy an ZIP-Dateien durchgeführt werden können

8.1 Benutzeroberfläche

Wenn eine ZIP-Datei in XMLSpy erstellt oder geöffnet wird, wird sie in der Archivansicht (*Abbildung unten)* geöffnet. Sie können gleichzeitig mehrere ZIP-Dateien öffnen, wobei jede ZIP-Datei in einem separaten Archivansichtsfenster angezeigt wird. Die Art der ZIP-Datei wird in der rechten oberen Ecke der Archivansicht angezeigt. In der Abbildung unten handelt es sich bei der ZIP-Datei um das Format MS Office Word Open XML.



Ordneransicht
Die Ordneransicht befindet sich auf der linken Seite des Archivansichtsfensters und zeigt die Ordnerstruktur des ZIP-Archivs an. Die Ordner auf den einzelnen Ebenen werden in alphabetischer Reihenfolge angezeigt. Um die Unterverzeichnisse eines Ordners zu sehen, klicken Sie auf das Plus-Symbol links vom Ordner. Wenn ein Ordner kein Plus-Symbol auf seiner linken Seite hat, so enthält er keine Unterverzeichnisse. Um die Dokumentdateien (in der Folge Dokumente genannt) in einem Ordner anzuzeigen, wählen Sie den Ordner aus. Die Dateien werden anschließend im Hauptfenster angezeigt. In der Abbildung oben befinden sich die im Hauptfenster angezeigten Dokumente im Ordner `word`. Dieser Ordner enthält zwei Unterverzeichnisse: `_rels` und `theme`.

Hauptfenster
Im Hauptfenster werden die Dokumente in dem Ordner aufgelistet, der in der Ordneransicht ausgewählt ist. Die Dokumente werden in alphabetischer Reihenfolge jeweils mit ihrer nicht komprimierten Größe und dem Datum und der Uhrzeit der letzten Änderung angezeigt. Um ein Dokument in der Archivansicht zu öffnen, doppelklicken Sie darauf. Das Dokument wird in einem separaten XMLSpy Fenster geöffnet.

Befehlsschaltflächen
Die Befehlsschaltflächen befinden sich entlang des oberen Rands des Archivansichtsfensters.

- ***Dokument öffnen:*** Ist aktiv, wenn ein Dokument im Hauptfenster ausgewählt ist. Wenn Sie darauf klicken, wird das ausgewählte Dokument geöffnet. Ein Dokument kann auch durch Doppelklick auf das im Hauptfenster aufgelistete Dokument geöffnet werden.
- ***Neuer Ordner:*** Fügt einen neuen Ordner zu dem Ordner hinzu, der in der Ordneransicht aktuell ausgewählt ist. Der Ordner muss sofort nach seiner Erstellung in der Ordneransicht benannt werden. Ein Ordner kann später nicht mehr umbenannt werden. Der neue Ordner wird beim Speichern der ZIP-Datei im Archiv gespeichert.
- ***Neues Dokument hinzufügen:*** Fügt zu dem gerade in der Ordneransicht ausgewählten Ordner ein neues Dokument hinzu. Das neue Dokument wird über das

XMLSpy-Dialogfeld "Neues Dokument erstellen" erstellt und in einem separaten XMLSpy Fenster geöffnet. Das Dokument muss unmittelbar, nachdem es in der Dokumentliste des ausgewählten Ordners aufgelistet wird, benannt werden. Das Dokument wird erst dann im Archiv gespeichert, wenn es in seinem Bearbeitungsfenster gespeichert wird oder wenn die ZIP-Datei gespeichert wird.

- ***Dokument hinzufügen:*** Öffnet ein Dialogfeld zum Durchsuchen, in dem Sie das gewünschte Dokument suchen können. Das Dokument wird zur Liste im Hauptfenster hinzugefügt, in dem die Dokumente im derzeit ausgewählten Ordner aufgelistet werden, und in einem separaten XMLSpy Fenster geöffnet. Damit das im Archiv gespeichert wird, muss es entweder im eigenen Fenster oder in der ZIP-Datei gespeichert werden.
- ***Aus Archiv löschen:*** Löscht das (im Hauptfenster) ausgewählte Dokument oder den ausgewählten Ordner (Ordneransicht) aus dem Archiv. Die ZIP-Datei muss gespeichert werden, damit die Dokumente tatsächlich gelöscht werden.
- ***Info:*** Blendet das Info-Fenster ein bzw. aus. *Siehe unten*.

Info-Fenster

Das Info-Fenster wird duch Klicken auf die Befehlsschaltfläche "Info" ein- und ausgeblendet. Das Info-Fenster enthält allgemeine Informationen zur ZIP-Datei, wie z.B. die Anzahl der darin enthaltenen Dateien, deren komprimierte und unkomprimierte Größe und das Komprimierungsverhältnis.

8.2 Operationen

Erstellen oder Öffnen einer ZIP-Datei
Wenn in XMLSpy eine neue Zip-Datei erstellt oder geöffnet wird, wird die Datei in der Archivansicht angezeigt.

- Neue ZIP-Dateien werden im XMLSpy Dialogfeld "Neues Dokument erstellen" (Befehl **Datei | Neu**) erstellt. In diesem Dialogfeld wählen Sie den gewünschten Dateityp aus und ein neues leeres Archiv wird in der Archivansicht geöffnet.
- Eine bestehende ZIP-Datei wird über das Dialogfeld "Öffnen" (**Datei | Öffnen**) geöffnet.

Bearbeiten der Struktur einer ZIP-Datei
Der Inhalt einer ZIP-Datei kann mit Hilfe der Funktionen der Benutzeroberfläche der Archivansicht durch Hinzufügen und Löschen von Ordnern und Dokumenten geändert werden. Nachdem Sie diese strukturellen Änderungen vorgenommen haben, muss die ZIP-Datei gespeichert werden, damit die Änderungen wirksam werden. Beachten Sie bitte die folgenden Punkte:

- Wenn ein neuer Ordner oder ein neues Dokument mit Hilfe der Befehlsschaltflächen der Benutzeroberfläche hinzugefügt wird, sollte er gleich bei der Erstellung benannt werden. Ein Ordner oder ein Dokument kann in der Archivansicht nicht umbenannt werden.
- Nachdem ein Dokument zu einem Ordner hinzugefügt wurde, wird es im Archiv gespeichert, indem es in seinem eigenen Fenster oder der Zip-Datei gespeichert wird.

Öffnen und Bearbeiten eines Dokuments
Ein Dokument wird über die Archivansicht geöffnet, indem Sie darauf doppelklicken oder es im Hauptfenster auswählen und auf die Befehlsschaltfläche Dokument öffnen klicken. Das Dokument wird in einem separaten XMLSpy Fenster geöffnet. Nachdem Sie es bearbeitet haben, speichern Sie das Dokument einfach wieder im ZIP-Archiv.

Bearbeiten von MS Office Open XML-Dokumenten
XMLSpy bietet intelligente Bearbeitungsfeatures für MS Word Open XML- und MS Excel Open XML-Dokumente. Diese Funktionen beinhalten Eingabehilfen, Autokomplettierungsfunktion und Validierung.

Pfadangaben für Dokumente in ZIP-Dateien
Die Pfadangaben für Dokumente in ZIP-Dateien sind normale Dateipfade plus das Pipe-Zeichen. z.B. wird mit dem Pfad:

```
C:\Program Files\Altova\XMLSpy2007\Examples\Office2007\ExcelDemo.xlsx|zip\xl\tables\table1.xml
```

die Datei `table1.xml` identifiziert, welche sich im Ordner `xl\tables` der ZIP-Datei `ExcelDemo.xlsx` im Ordner `Examples\Office2007` befindet des XMLSpy Applikationsordners befindet.

Beispieldateien
Im Ordner `Examples\Office2007` Ihres XMLSpy Applikationsordners befinden sich die folgenden Beispieldateien:

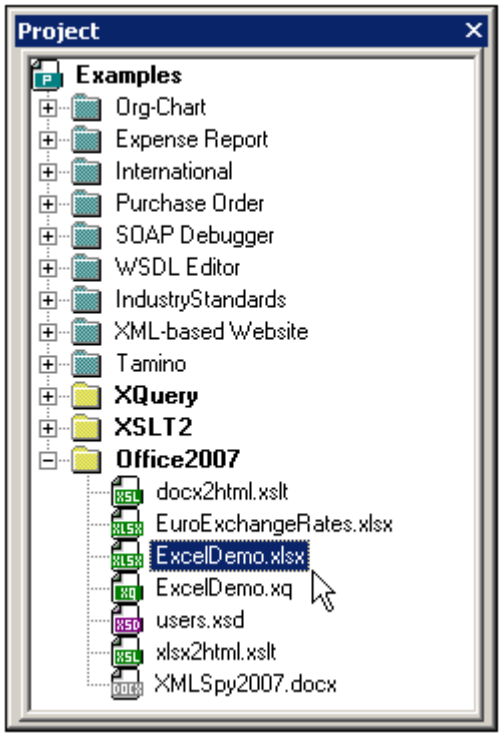
- ZIP-Dateien: (i) eine MS Word Open XML-Datei (`.docx`), und (ii) eine MS Excel Open XML-Datei (`.xlsx`)
- XSLT-Dateien: (i) `docx2html.xslt` (zum Konvertieren der `.docx` Beispieldatei in

© 2007 Altova GmbH

HTML), und (ii) `xslx2html.xslt` (zum Konvertieren der `.xslx` Beispieldatei in HTML)
- Eine XQuery-Datei: `ExcelDemo.xq` (zum Abrufen der Daten aus der `.xslx` Datei)

Die XSLT- und die XQuery-Datei sind dazu gedacht, um zu zeigen, wie man mit Hilfe von XSLT und XQ Daten in ZIP-Archiven aufrufen und transformieren kann. Zum Ausführen dieser Dokumente können Sie eine der folgenden Optionen verwenden:

- Öffnen Sie die ZIP-Datei in der Archivansicht. Wählen Sie in der Ordneransicht `Archive` aus und klicken Sie auf den Menübefehl **XSL/XQuery | XSL Transformation** (um eine XSLT-Transformation durchzuführen) oder **XSL/XQuery | XQuery-Ausführung** (um eine XQuery auszuführen). Suchen Sie die XSLT- oder die XQuery-Datei und klicken Sie auf **OK**.
- Klicken Sie im Projektfenster von XMLSpy im Ordner `Office2007` des Projekts `Examples` mit der rechten Maustaste auf die Datei `.xlsx` oder `.docx` (*Abbildung unten*) und wählen Sie den Transformationsbefehl. Suchen Sie die Transformationsdatei und klicken Sie auf **OK**.

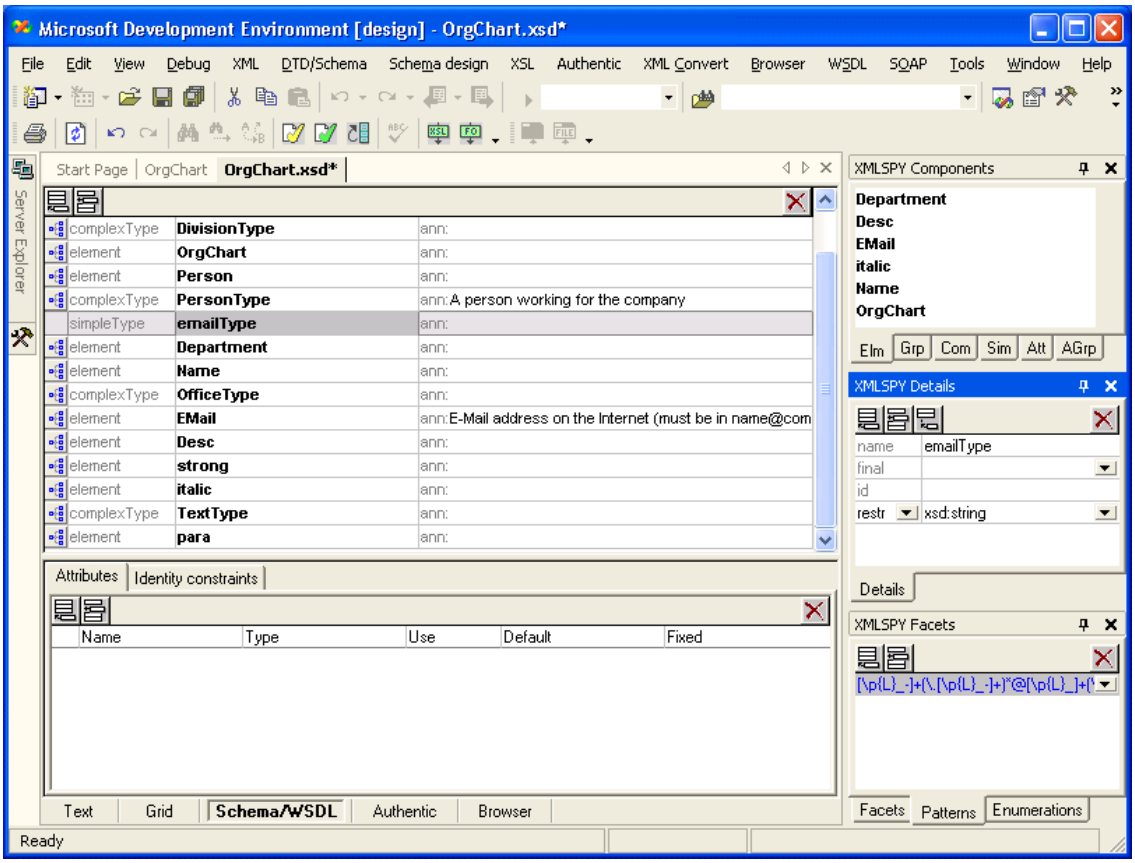


- Öffnen Sie die XSLT oder XQuery-Datei in XMLSpy und klicken Sie auf den Menübefehl **XSL/XQuery | XSL-Transformation** bzw. **XSL/XQuery | XQuery-Ausführung**. Wenn Sie aufgefordert werden, die gewünschte XML-Datei anzugeben, suchen Sie nach der `.docx-` oder `.xlsx`-Datei (je nachdem, ob das XSLT/XQ Dokument für Word oder Excel gedacht ist).
- Sie können diese Operationen mittels AltovaXML, einem eigenständigen Paket, das die Altova XSLT- und XQuery-Prozessoren enthält, ausführen.

9 XMLSpy in MS Visual Studio .NET

Sie können Ihre XMLSpy Version in die Microsoft Visual Studio .NET IDE-Versionen 2002 und 2003 integrieren. Auf diese Weise stehen Ihnen sowohl die hoch entwickelten XML-Editierfunktionen als auch die professionelle Entwicklungsumgebung von Visual Studio .NET zur Verfügung:

- Installieren Sie Microsoft Visual Studio .NET
- Installieren Sie XMLSpy (Enterprise oder Professional Edition)
- Laden Sie das XMLSpy Integrationspaket für Microsoft Visual Studio .NET herunter und starten Sie es. Sie finden das Installationspaket auf der XMLSpy (Enterprise und Professional Edition) Download-Seite unter www.altova.com. (**Bitte beachten Sie:** Sie müssen das Integrationspaket für Ihre jeweilige Version von XMLSpy (Enterprise oder Professional) verwenden).



Nach Installation des Integrationspakets können Sie XMLSpy innerhalb der Visual Studio .NET-Umgebung verwenden.

© 2007 Altova GmbH

9.1 Unterschiede zwischen der .NET und der Standalone-Version

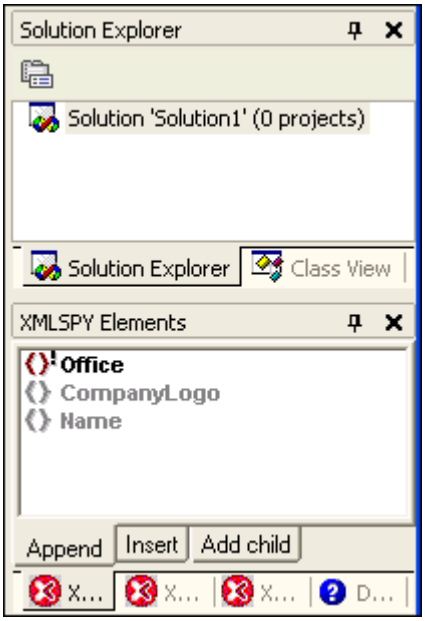
In diesem Abschnitt wird beschrieben, inwiefern sich die Visual Studio .NET Versionen von den Standalone-Versionen von XMLSpy unterscheiden. Einige Funktionen, welche in den Standalone-Versionen, nicht aber in den Visual Studio .Net Versionen zur Verfügung stehen, sind am Ende dieses Abschnitts aufgelistet.

XMLSpy-Menüs oder -Menüoptionen, die beim Start von Visual Studio .NET angezeigt werden:

- File | New | AUTHENTIC
- XML-Konvertierung
- SOAP
- Tools | XMLSpy Optionen

Eingabehilfen (Toolfenster in Visual Studio .NET)

- Die Eingabehilfen-Fenster werden beim Start unterhalb des Projektmappen-Explorers auf einer Registerkarte gruppiert.
- Sie können die Eingabehilfen-Fenster an jede beliebige Stelle in der Entwicklungsumgebung ziehen.
- Wenn Sie auf ein Eingabehilfen-Register rechtsklicken, können Sie die Schnittstelle weiter anpassen. Sie können die Eingabehilfen-Fenster als abgedockt, angedockt, ausgeblendet und automatisch ausgeblendet konfigurieren.



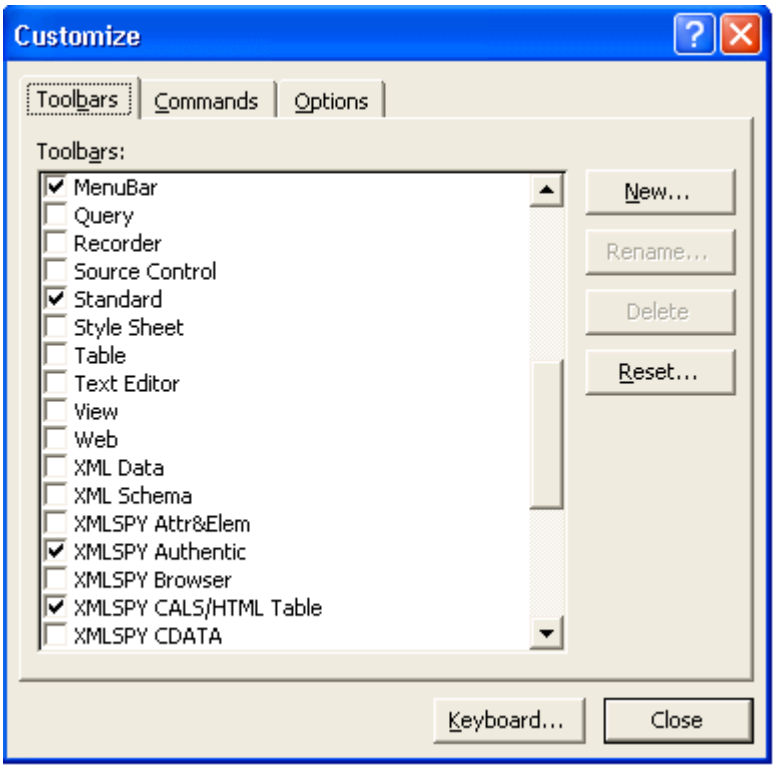
Geänderte Funktionen in den Visual Studio .NET Editions:

File-Menü
Verwenden Sie die Befehle **File | Open | File from web** anstelle von **File | URL öffnen** um eine Datei über eine URL zu öffnen.
Verwenden Sie die Befehle **File | Save "my file" as | My network places** anstelle von **File | Speichern unter URL** um eine Datei unter einer URL zu speichern.

Edit-Menü **,** Undo und Redo
Die Befehle Undo und Redo wirken sich auf alle Aktionen (Kopieren, Einfügen usw.) aus, die in der Entwicklungsumgebung und in XMLSpy durchgeführt wurden.

Bitte beachten Sie, dass es zwei Symbolsätze zum Rückgängigmachen und Wiederherstellen einer Aktion gibt. Die XMLSpy-Symbole befinden sich in der XMLSpy-Hauptsymbolleiste. Die Visual Studio .NET-Symbole sind deaktiviert, solange Dateien in XMLSpy geöffnet sind.

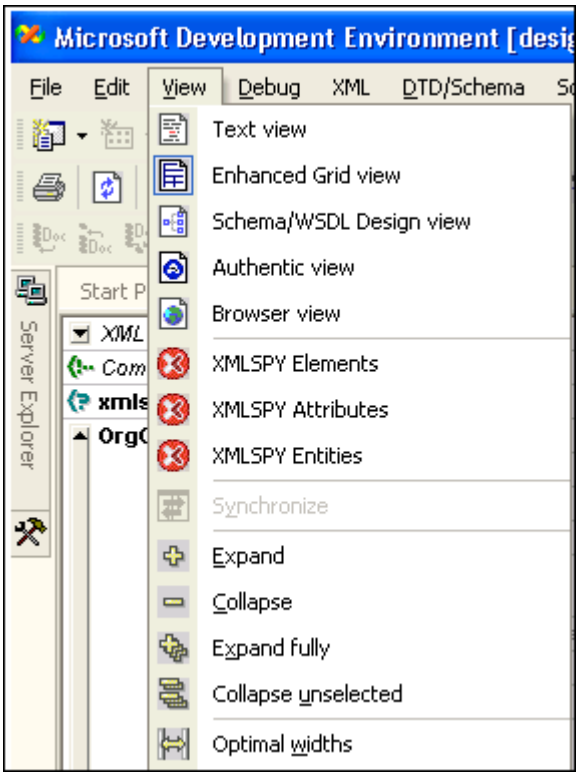
Menü **Tools | Customize | Toolbar**, **Commands** und **Options**.



Diese Registerkarten enthalten sowohl Visual Studio .NET- als auch XMLSpy-Befehle.

View-Menü

© 2007 Altova GmbH



Im Menü View (Ansicht) können Sie Folgendes auswählen bzw. aufrufen:

- die Art der Ansicht, in der Ihr XML-Dokument angezeigt werden soll: Text-, Enhanced Grid-, Schema/WSDL-, Authentic- oder Browser-Ansicht.
- eventuell zuvor geschlossene XMLSpy-Eingabehilfen-Fenster.
- XSLT- und SOAP Debugger Tool-Fenster, die unter Umständen während eines Debugger-Vorgangs geschlossen wurden.
- Funktionen der Grid-Ansicht zur übersichtlicheren Anzeige von Daten.

Help-Menü
Das Hilfemenü enthält das Untermenü XMLSpy Hilfe, über das Sie die XMLSpy-Hilfe aufrufen können. Des Weiteren finden Sie darin Links zum Altova Support Center, einen Bereich zum Downloaden von Komponenten usw.

Nicht unterstützte Features der .NET Edition von XMLSpy

Info-Fenster
Das Info-Fenster wird nicht unterstützt. Dieses Fenster enthält zusätzliche Informationen zu ausgewählten Elementen/Attributen usw., d.h. es enthält den Namen des Elements/Attributs sowie Informationen zu datatype, enumeration und occurrence.

Projekt-Fenster und **Projekt-**Menü (sowie die Versionskontroll-Untermenüs).
Das Projekt-Menü (und die Versionskontroll-Untermenüs) stehen nicht zur Verfügung, da MS Visual Studio .NET seine eigene Projekt- und Versionskontroll-Umgebung hat. Aus diesem Grund werden die Validierung und Umwandlung von Dateien im Batch-Modus derzeit nicht unterstützt.

Extras | Skripting-Umgebung...
Die Skripting-Umgebung steht derzeit nicht zur Verfügung.

Eigenes Browser-Fenster
Das Kontrollkästchen "In eigenem Fenster anzeigen" im Menü **Extras | Optionen |** Register

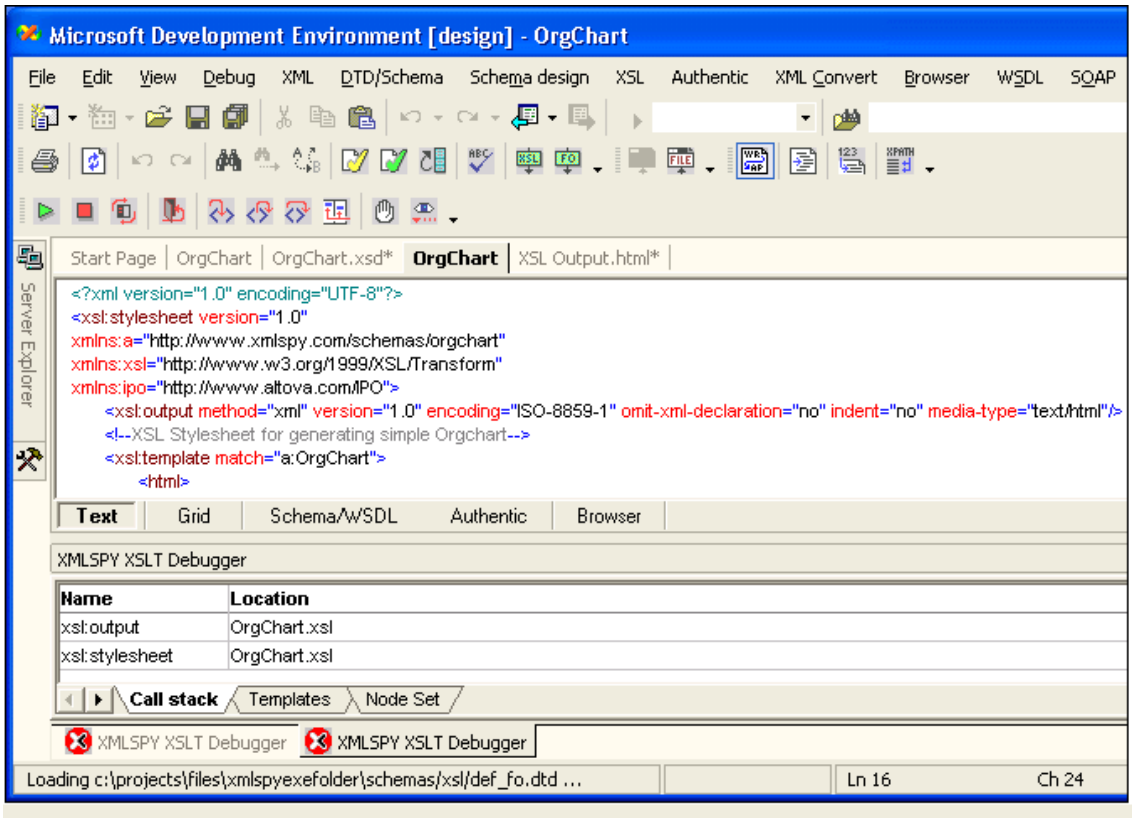
Ansicht wird nicht unterstützt, daher werden die Text- und die Browser-Ansicht immer in ein und demselben Fenster angezeigt - z.B. beim Transformieren einer XML-Datei in eine HTML-Datei.

Authentic-Ansicht
Symbole zum Textstatus sind in der Authentic-Ansicht nicht verfügbar. Die Funktionen bzw. die Formatierung, die Sie damit aktivieren, können jedoch weiterhin über die Eingabehilfe oder das Kontextmenü aufgerufen werden. Nähere Informationen dazu finden Sie in der Dokumentation zu Stylesheet Designer.

© 2007 Altova GmbH

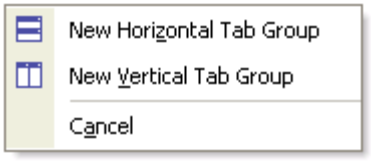
9.2 Visual Studio .NET und XSLT Debugger

Im unten gezeigten Screenshot sehen Sie, wie die Datei `OrgChart.xml` (im Ordner `Examples`) angezeigt wird, wenn der XSLT Debugger aktiviert ist. Standardmäßig werden alle Dateien auf einer Registerkartengruppe angezeigt. Um den Debugging-Vorgang übersichtlicher zu gestalten, können Sie in Visual Studio .NET eine eigene Registerkarte anlegen.

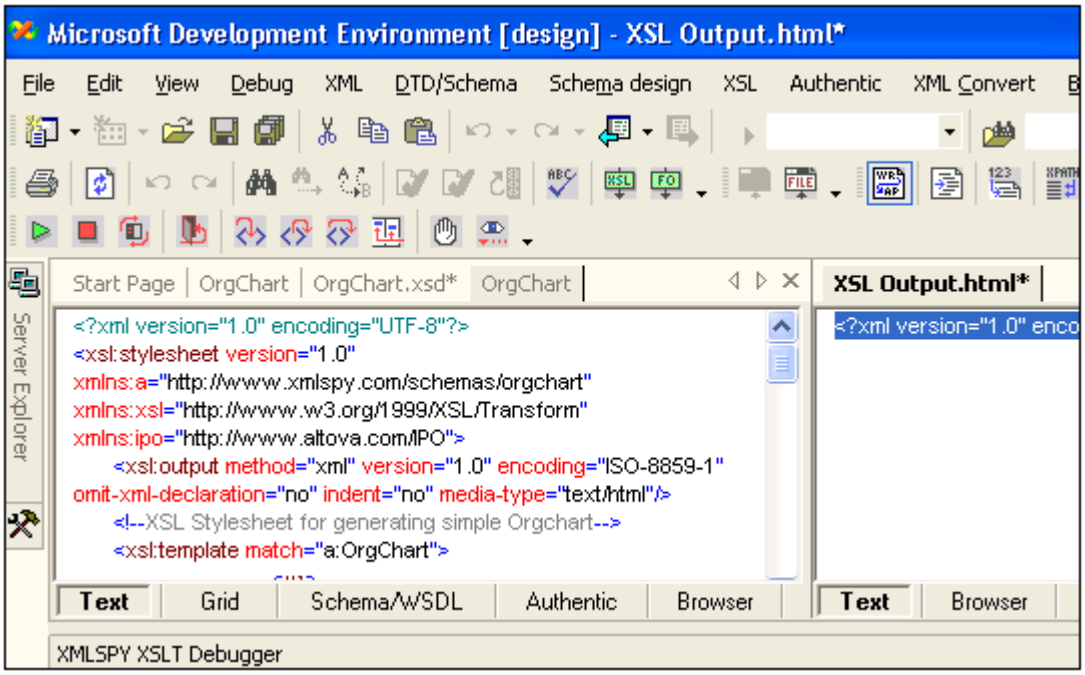


So legen Sie eine neue Registerkartengruppe an:

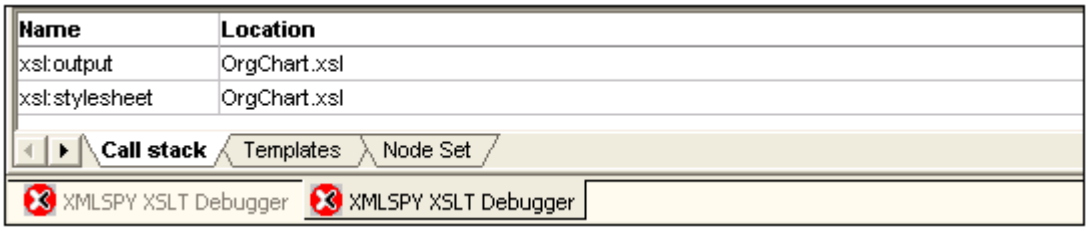
1. Klicken Sie auf das Register **XSL Output.html** und ziehen Sie es auf die aktive Registerkarte. Hierauf wird ein Popup-Menü angezeigt, in dem Sie die Art der zu erstellenden Registerkarte definieren können.



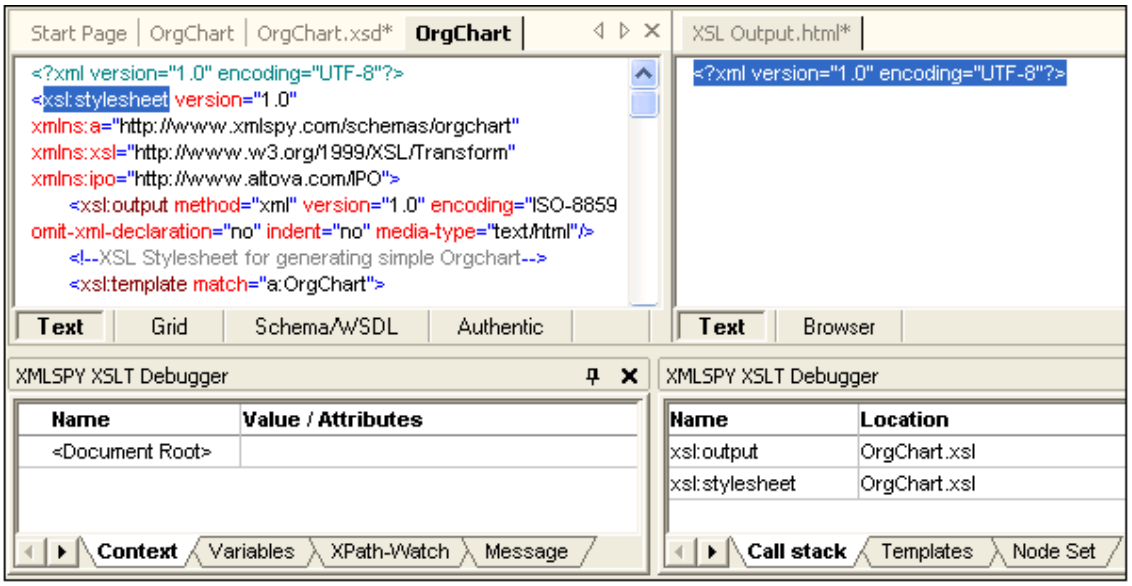
2. Wählen Sie die Option **New Vertical Tab Group**.Daraufhin wird eine neue Registerkarte mit der Datei `XSLOutput.html` erstellt.



3. Erstellen Sie mit derselben Methode im Debugger-Fenster eine neue Registerkartengruppe.

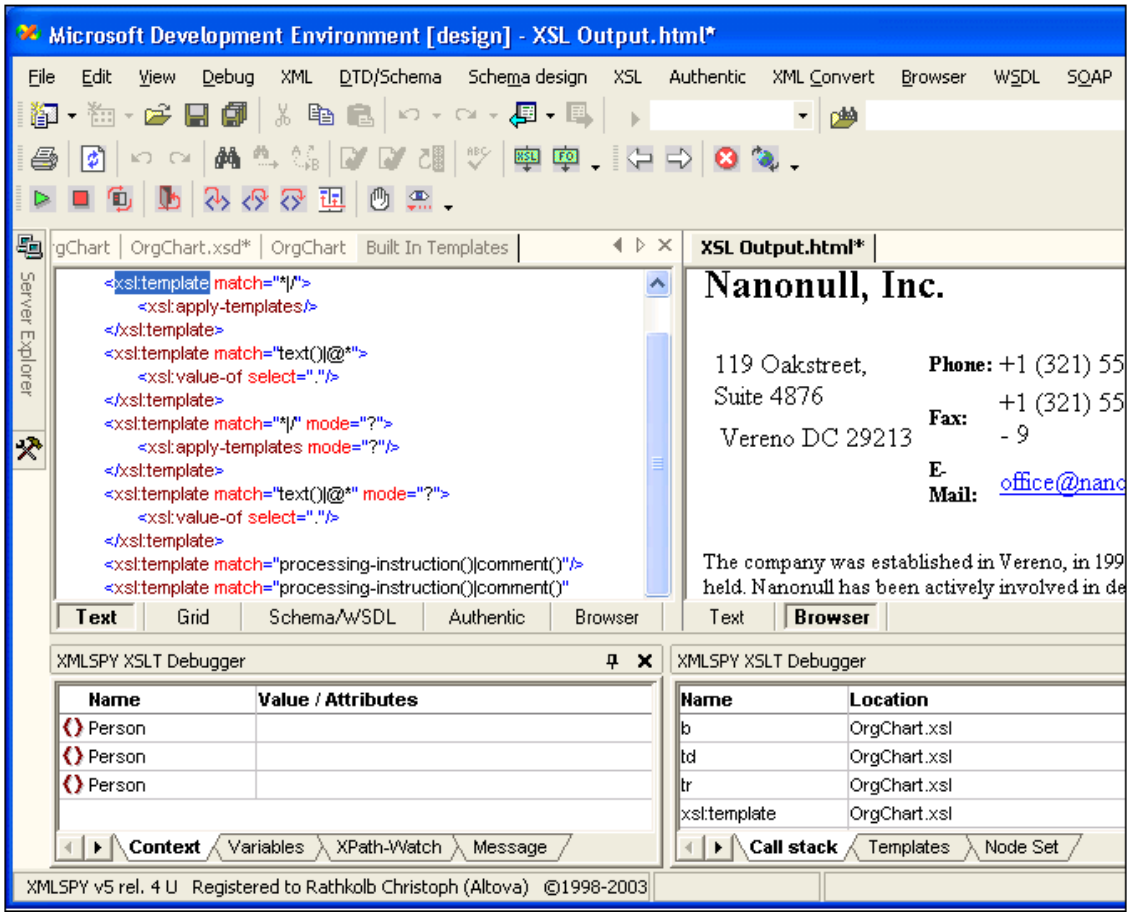


4. Ziehen Sie das ganz rechts angezeigte **XMLSPY XSLT Debugger**-Fenster nach rechts.
5. Wählen Sie im Popup-Menü **New Vertical Tab Group**.



 © 2007 Altova GmbH

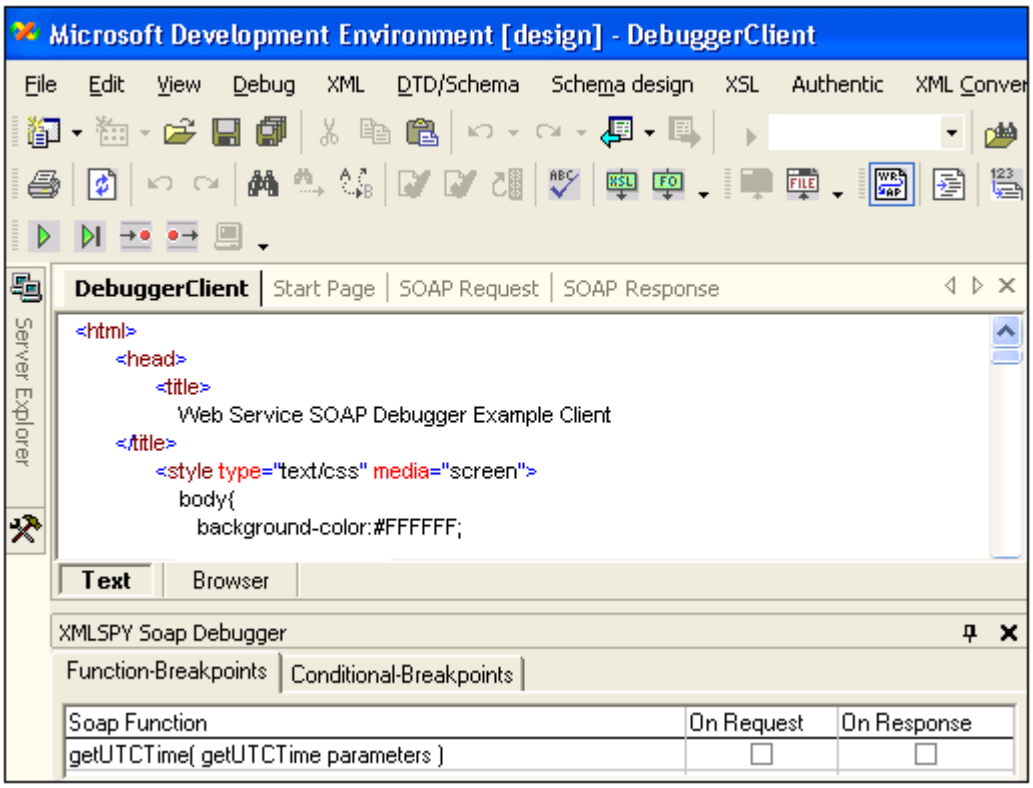
Sie können die Datei OrgChart.xml nun während des Debugging-Vorgangs im XSLT-Debugger anzeigen:



9.3 Visual Studio .NET und SOAP Debugger

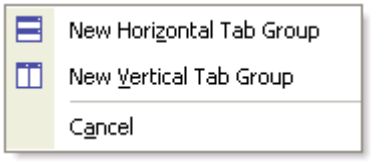
SOAP Debugger

Im unten gezeigten Screenshot sehen Sie, wie die Datei DebuggerClient.html angezeigt wird, wenn der SOAP Debugger aktiviert wurde. Standardmäßig werden alle SOAP-Fenster auf einer Registerkartengruppe angezeigt. Um den Debugging-Vorgang übersichtlicher zu gestalten, können Sie Ihre eigene Registerkartengruppe in Visual Studio .NET anlegen.



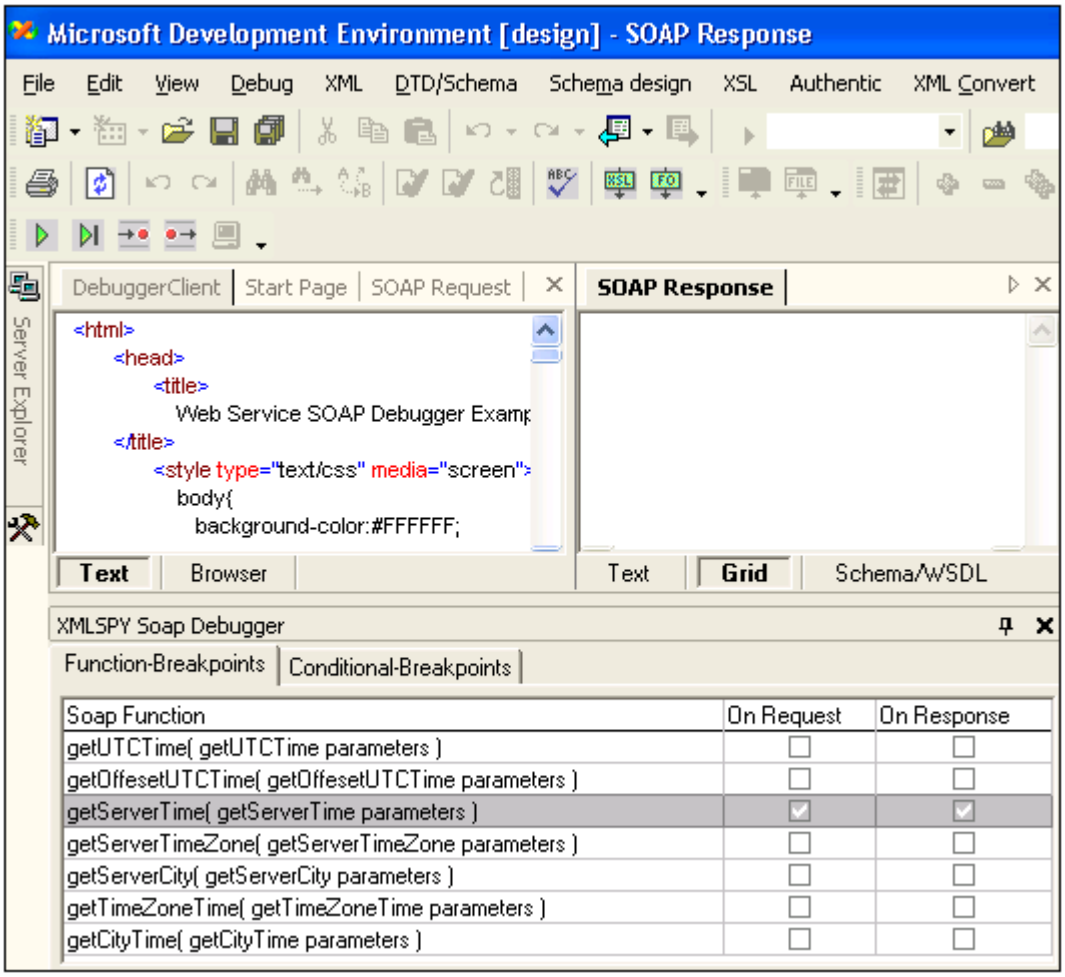
So legen Sie eine neue Registerkartengruppe an:

1. Klicken Sie auf das Register **SOAP Response** und ziehen Sie es auf die aktive Registerkarte.
2. Hierauf wird ein Popup-Menü angezeigt, in dem Sie die Art der zu erstellenden Registerkarte definieren können.



3. Wählen Sie die Option "New Vertical Tab Group".
   Daraufhin wird eine neue Registerkarte mit dem SOAP Response-Fenster erstellt.

© 2007 Altova GmbH


Microsoft Development Environment [design] - SOAP Response
File Edit View Debug XML DTD/Schema Schema design XSL Authentic XML Convert
DebuggerClient Start Page SOAP Request
SOAP Response
<html>
<head>
<title>
Web Service SOAP Debugger Examp
</title>
<style type="text/css" media="screen">
body{
background-color:#FFFFFF;
Text Browser
Text Grid Schema/WSDL
Server Explorer
XMLSPY Soap Debugger
Function-Breakpoints Conditional-Breakpoints
Soap Function
On Request
On Response
getUTCTime(getUTCTime parameters)
getOffesetUTCTime(getOffesetUTCTime parameters)
getServerTime(getServerTime parameters)
getServerTimeZone(getServerTimeZone parameters)
getServerCity(getServerCity parameters)
getTimeZoneTime(getTimeZoneTime parameters)
getCityTime(getCityTime parameters)

10 XMLSpy in Eclipse

Eclipse 3.x ist eine Open-Source-Umgebung, in die verschiedene Arten von Applikationen in Form von Plug-Ins integriert werden können. XMLSpy für die Eclipse-Plattform ist ein Eclipse Plug-In, das Ihnen gestattet, auf die Funktionalitäten einer zuvor installierten Version von XMLSpy Edition über die Eclipse 3.0 / 3.1 und 3.2-Plattform zuzugreifen.

Voraussetzungen für die Installation

Um das XMLSpy Plug-In für Eclipse 3.x installieren zu können, benötigen Sie:

- die XMLSpy Edition, die Sie verwenden möchten: Enterprise, Professional oder Home
- das Eclipse 3.x-Paket sowie
- die entsprechende Java Runtime-Version

Installation des XMLSpy-Plugin

So installieren Sie das XMLSpy-Plugin:

- Laden Sie das XMLSpy Plugin für Eclipse vom Download-Bereich der Altova.com Website herunter. Während der Installation werden Sie aufgefordert, den Installationsordner für das Plug-In anzugeben.

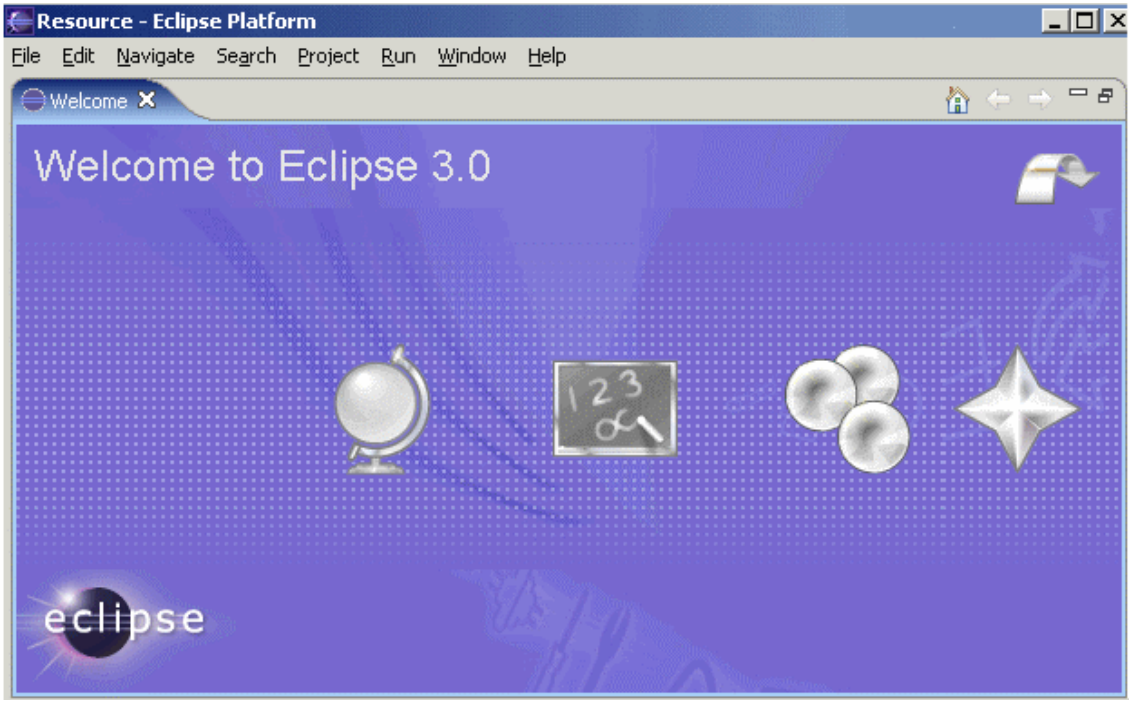
Das XMLSpy Plug-In für Eclipse bietet die folgenden Funktionalitäten:

- einen Editor mit vollem Funktionsumfang, mit dem Sie jede Dateiart, die XMLSpy unterstützt, bearbeiten können. Die applikationsspezifischen Befehle werden zur Menü- und Symbolleiste hinzugefügt.
- eine Reihe von Ansichten, die die einzelnen Anwendungsfenster enthalten: in diesem Fall die XMLSpy Eingabehilfen sowie die einzelnen spezifischen Debuggerfenster.
- verschiedene Perspektiven, die das Aussehen der Workbench bestimmen. XMLSpy bietet drei Perspektiven: XMLSpy, Debug SOAP und Debug XSLT.
- eine XMLSpy Benutzerhilfe unter dem Menüeintrag **Help | XMLSPY |** Inhaltsverzeichnis.

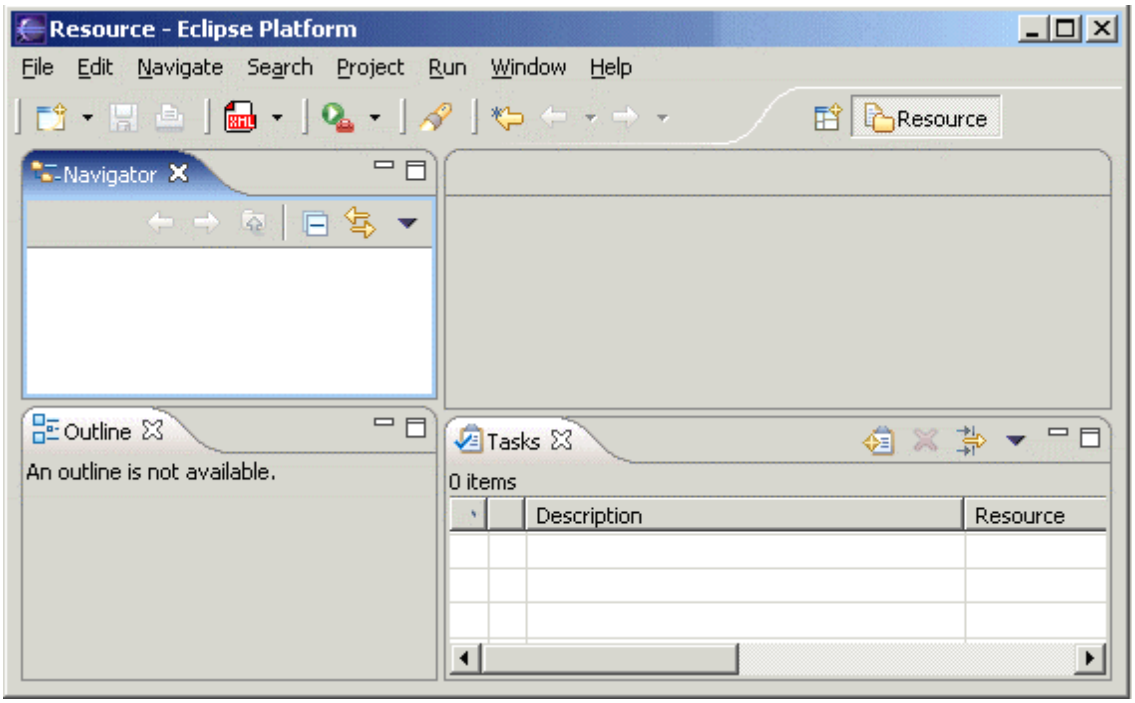
© 2007 Altova GmbH

10.1 Starten von Eclipse und Verwendung des XMLSpy Plug-Ins

1. Doppelklicken Sie auf eclipse.exe, um die Eclipse-Plattform zu starten.

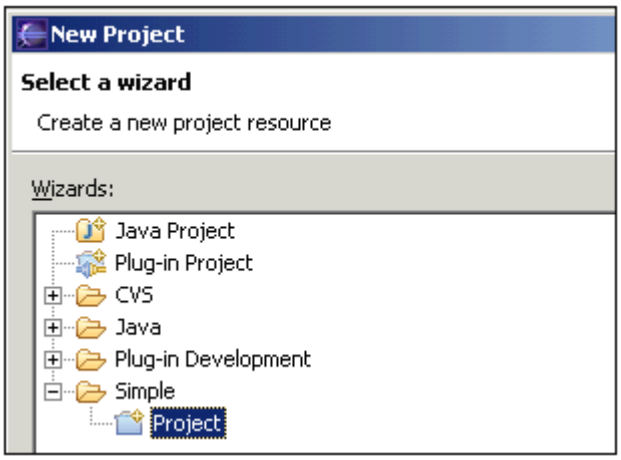


   Daraufhin erscheint der Willkommensbildschirm von Eclipse 3.0
2. Platzieren Sie den Cursor über das Pfeilsymbol und klicken Sie, wenn der Text "Go To Workbench" angezeigt wird.
3. Daraufhin wird ein leeres XMLSpy-Fenster in Eclipse geöffnet.

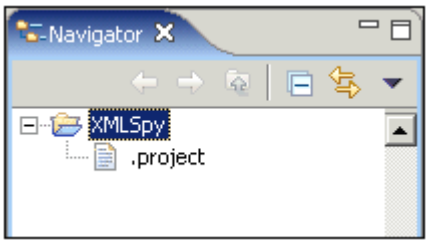


Erstellen eines neuen Projekts:

1. Rechtsklicken Sie in das Navigatorfenster und wählen Sie **New | Project | Simple Project**.



2. Geben Sie XMLSpy als Projektnamen ein und klicken Sie auf "Finish".
3. Daraufhin wird der XMLSpy-Projektordner erstellt.

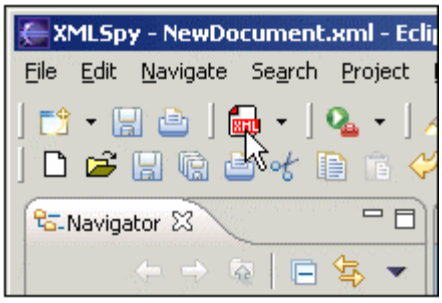


© 2007 Altova GmbH

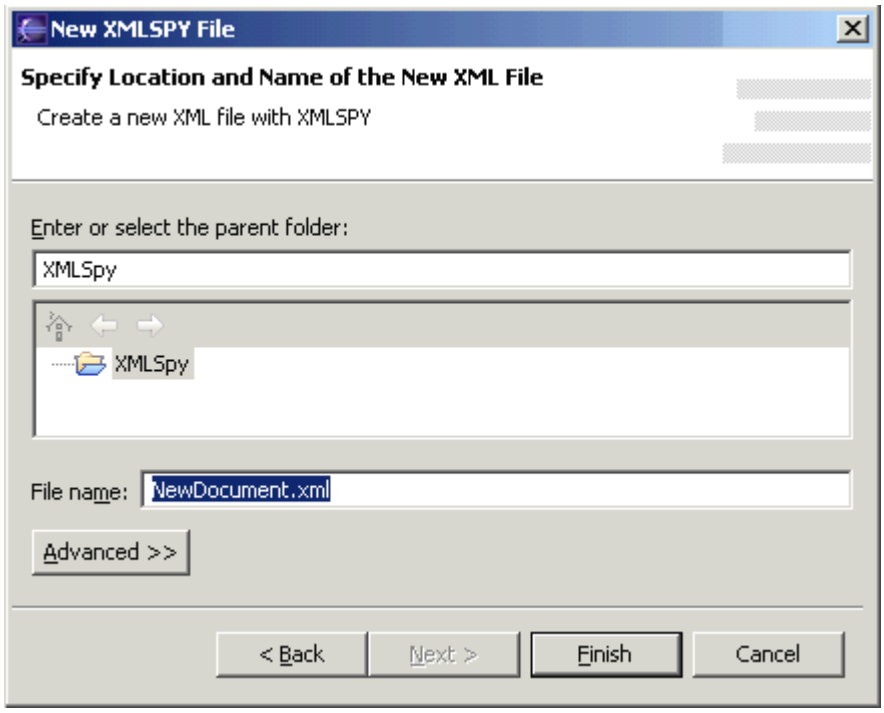
10.2 Erstellen von XML-Dateien in Eclipse

So erstellen Sie eine neue XML-Datei auf Basis eines Schemas:

1. Klicken Sie auf das XMLSpy XML-Symbol.

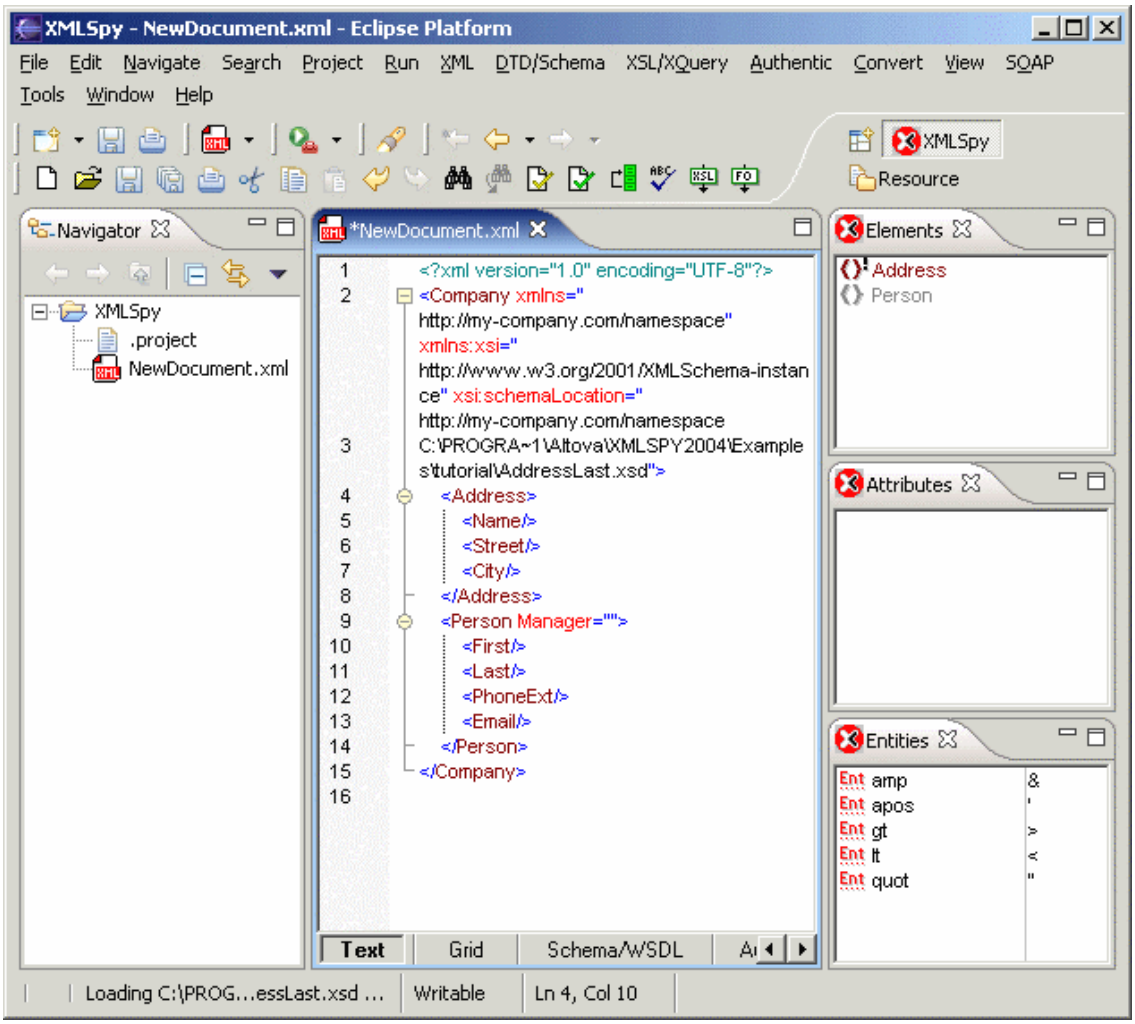


2. Wählen Sie **xml | XM-Dokument** und klicken Sie auf "Next".
   Sie werden nun aufgefordert, einen übergeordneten Ordner auszuwählen.



3. Geben Sie XMLSpy als übergeordneten Ordner (Parent folder) an (oder wählen Sie einen vorhandenen Ordner aus), verwenden Sie den vorgeschlagenen Standard-Dateinamen **NeuesDokument.xml** und klicken Sie auf "Finish".
   Die Dialogfelder, die nun angezeigt werden, stammen aus XMLSpy.
4. Wählen Sie Schema und klicken Sie auf OK, wählen Sie anschließend mit Hilfe der Durchsuchen-Schaltfläche die Schemadatei (z.B. AddressLast.xsd) und klicken Sie auf OK.
   Die neue XML-Datei wird nun in der Textansicht auf dem Register NeuesDokument.xml angezeigt.

   Die vorkonfigurierte XMLSpy-Perspektive wird automatisch aktiviert, um die verschiedenen Eingabehilfen anzuzeigen.

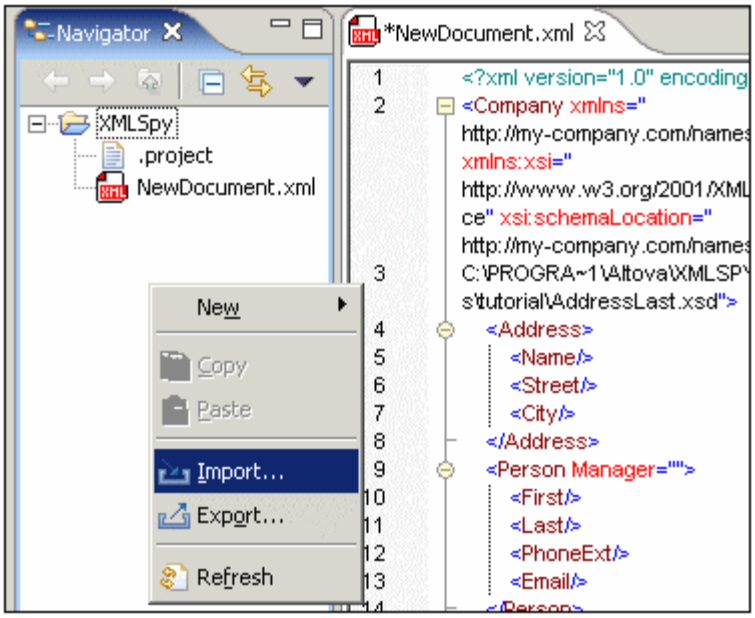


© 2007 Altova GmbH

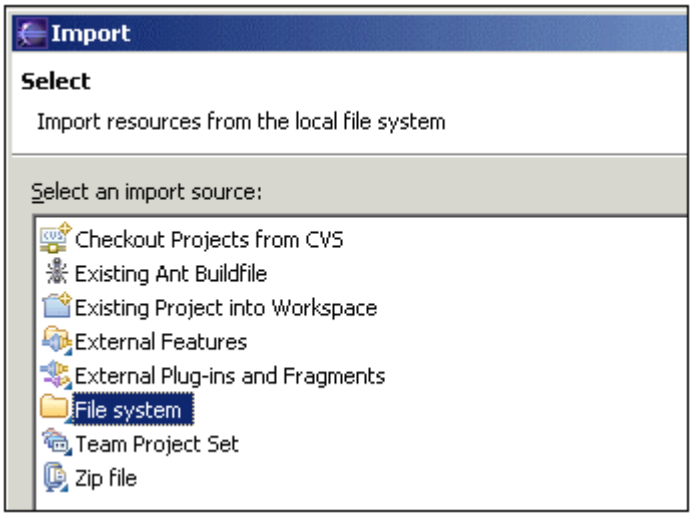
10.3 Importieren von XML-Dateien in Eclipse-Ordner

Importieren des XMLSpy Ordners "Examples" in den Navigator:

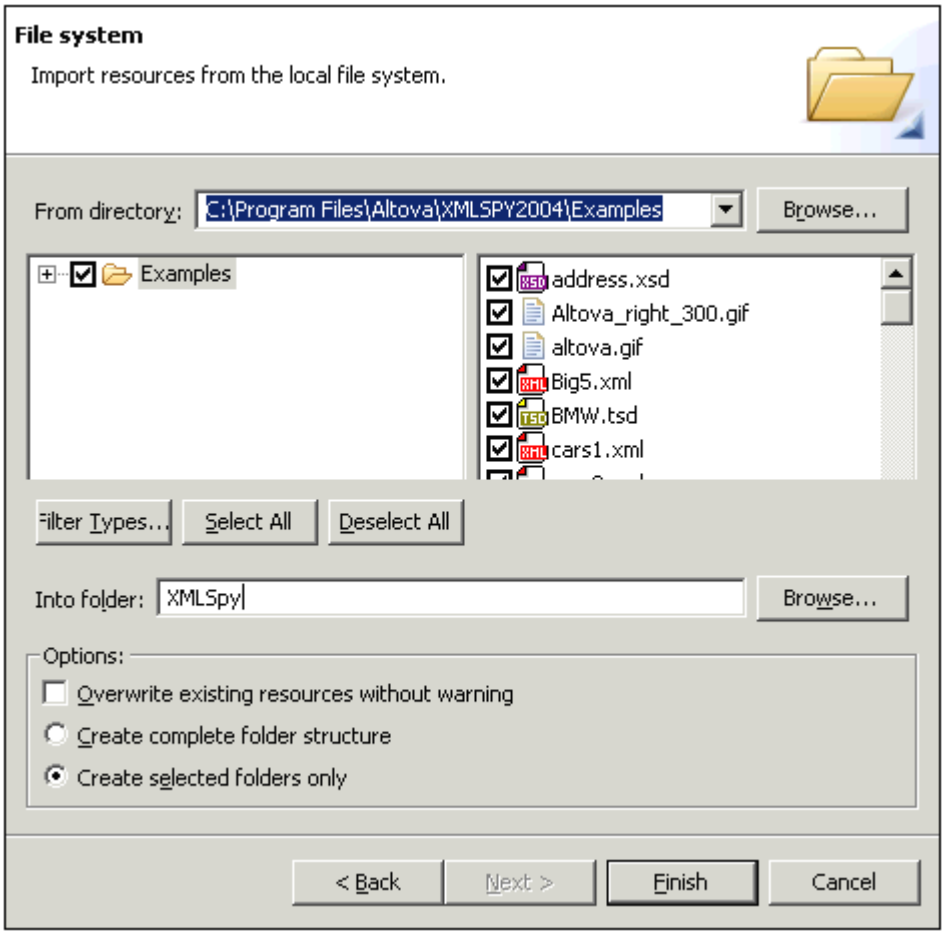
1. Rechtsklicken Sie auf das Register "Navigator" und wählen Sie "Import".



2. Wählen Sie "File system" und klicken Sie auf "Next".

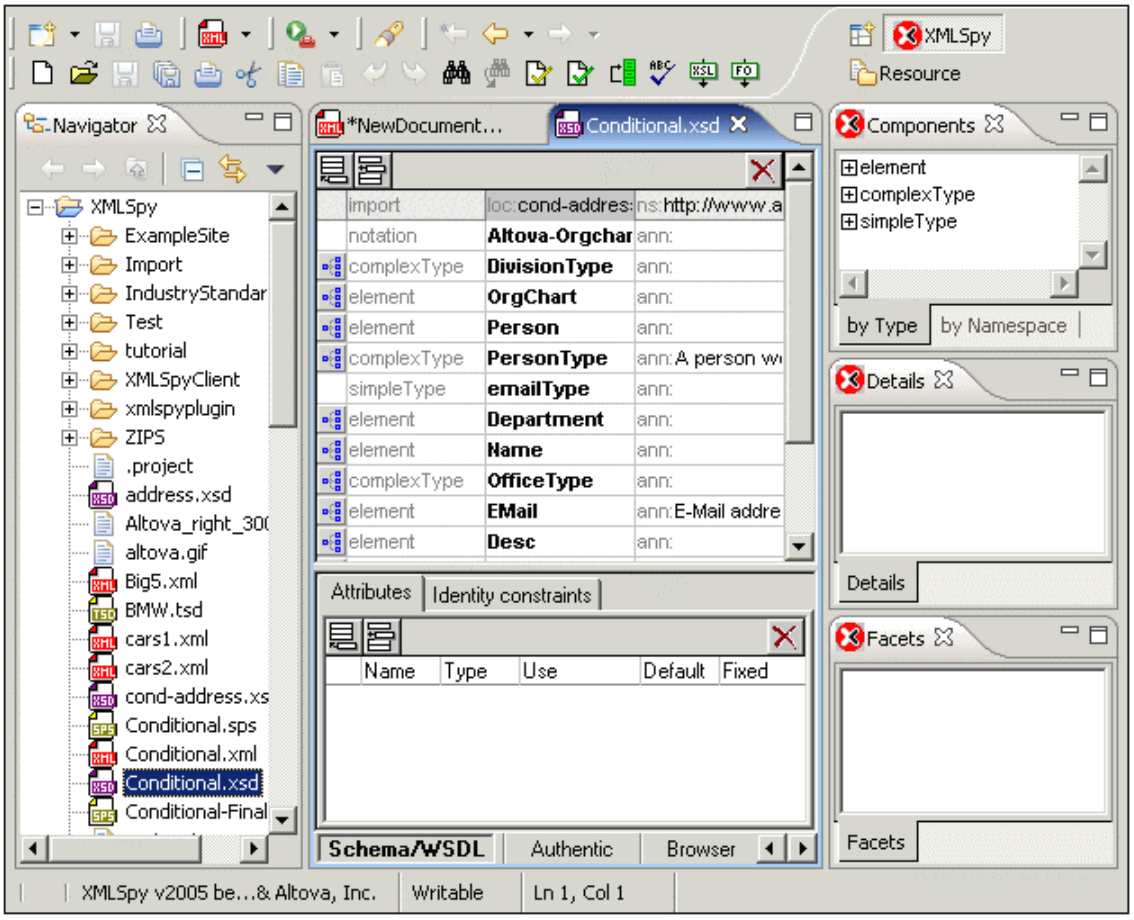


3. Klicken Sie auf die Schaltfläche "Browse" rechts vom Textfeld "From directory:" und wählen Sie das Verzeichnis "Examples" in Ihrem XMLSpy-Ordner aus.



4. Aktivieren Sie das Kontrollkästchen **Examples**.
   Daraufhin werden alle Dateien in den verschiedenen Unterverzeichnissen im Fenster auf der rechten Seite aktiviert.
5. Klicken Sie auf die Schaltfläche "Browse" neben dem Textfeld "Into folder:" und wählen Sie den Zielordner aus. Klicken Sie anschließend auf "Finish".
   Die ausgewählten Ordner und Dateien werden in den Eclipse-Arbeitsbereich kopiert.
6. Doppelklicken Sie auf eine Datei im Navigator, um sie zu öffnen (z.B. Conditional.xsd).

© 2007 Altova GmbH


XMLSpy
Resource
Navigator
*NewDocument...
Conditional.xsd
Components
element
complexType
simpleType
by Type
by Namespace
Details
Facets
XMLSpy
ExampleSite
Import
IndustryStandar
Test
tutorial
XMLSpyClient
xmlspyplugin
ZIPS
.project
address.xsd
altova.gif
Big5.xml
BMW.tsd
cars1.xml
cars2.xml
cond-address.xs
Conditional.sps
Conditional.xml
Conditional.xsd
Conditional-Final
import
notation
complexType
element
simpleType
DivisionType
OrgChart
Person
PersonType
emailType
Department
Name
OfficeType
EMail
Desc
Attributes
Identity constraints
Name
Type
Use
Default
Fixed
Schema/WSDL
Authentic
Browser
XMLSpy v2005 be...& Altova, Inc.
Writable
Ln 1, Col 1

10.4 Unterschiede zwischen Eclipse und den Standalone-Versionen

Die Enterprise, Professional und Home Edition des Eclipse Plug-Ins für XMLSpy haben im Allgemeinen dieselben Funktionalitäten wie die jeweiligen Standalone-Versionen.

Nicht unterstützte Funktionen in den integrierten Versionen:

Info-Fenster
Das Info-Fenster wird nicht unterstützt. Dieses Fenster enthält zusätzliche Informationen zu ausgewählten Elementen/Attributen usw., d.h. es enthält den Namen des Elements/Attributs sowie Informationen zu datatype, enumeration und occurrence.

Projekt-Fenster und **Projekt-**Menü (sowie die Versionskontroll-Untermenüs).
Das Projekt-Menü (und die Versionskontroll-Untermenüs) stehen nicht zur Verfügung. Verwenden Sie stattdessen den Eclipse Navigator.

Extras | Skripting-Umgebung...
Die Skripting-Umgebung steht derzeit nicht zur Verfügung.

Eigenes Browser-Fenster
Das Kontrollkästchen "In eigenem Fenster anzeigen" im Dialogfeld **Optionen (Extras | Optionen,** Register **Ansicht)** wird nicht unterstützt, daher werden die Text- und die Browser-Ansicht immer in ein und demselben Fenster angezeigt - z.B. beim Transformieren einer XML-Datei in eine HTML-Datei.

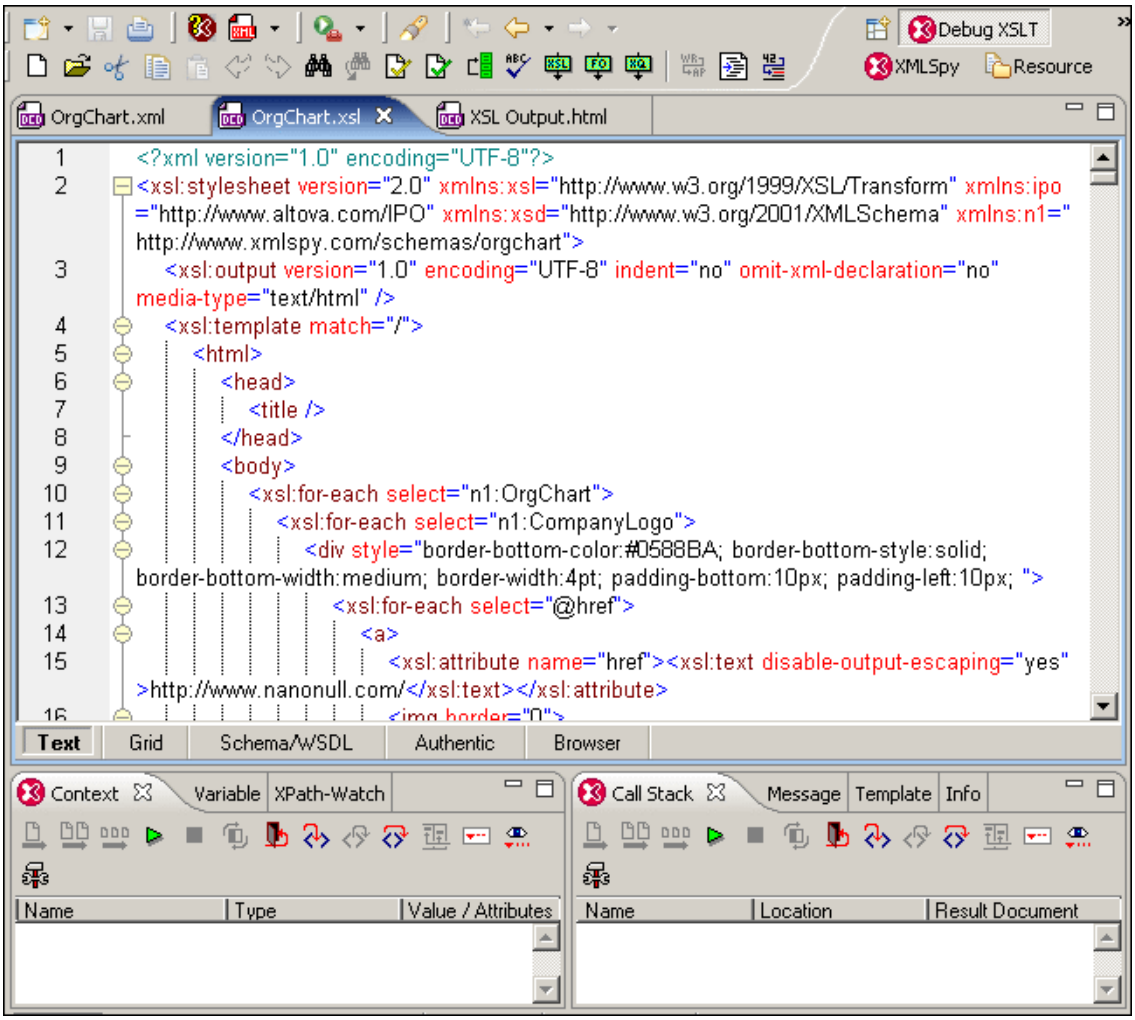
Authentic-Ansicht
Symbole zum Textstatus sind in der Authentic-Ansicht nicht verfügbar. Die Funktionen bzw. die Formatierung, die Sie damit aktivieren, können jedoch weiterhin über die Eingabehilfe oder das Kontextmenü aufgerufen werden. Nähere Informationen dazu finden Sie in der Dokumentation zu Stylesheet Designer.

© 2007 Altova GmbH

10.5 Eclipse-Plattform und XSLT Debugger

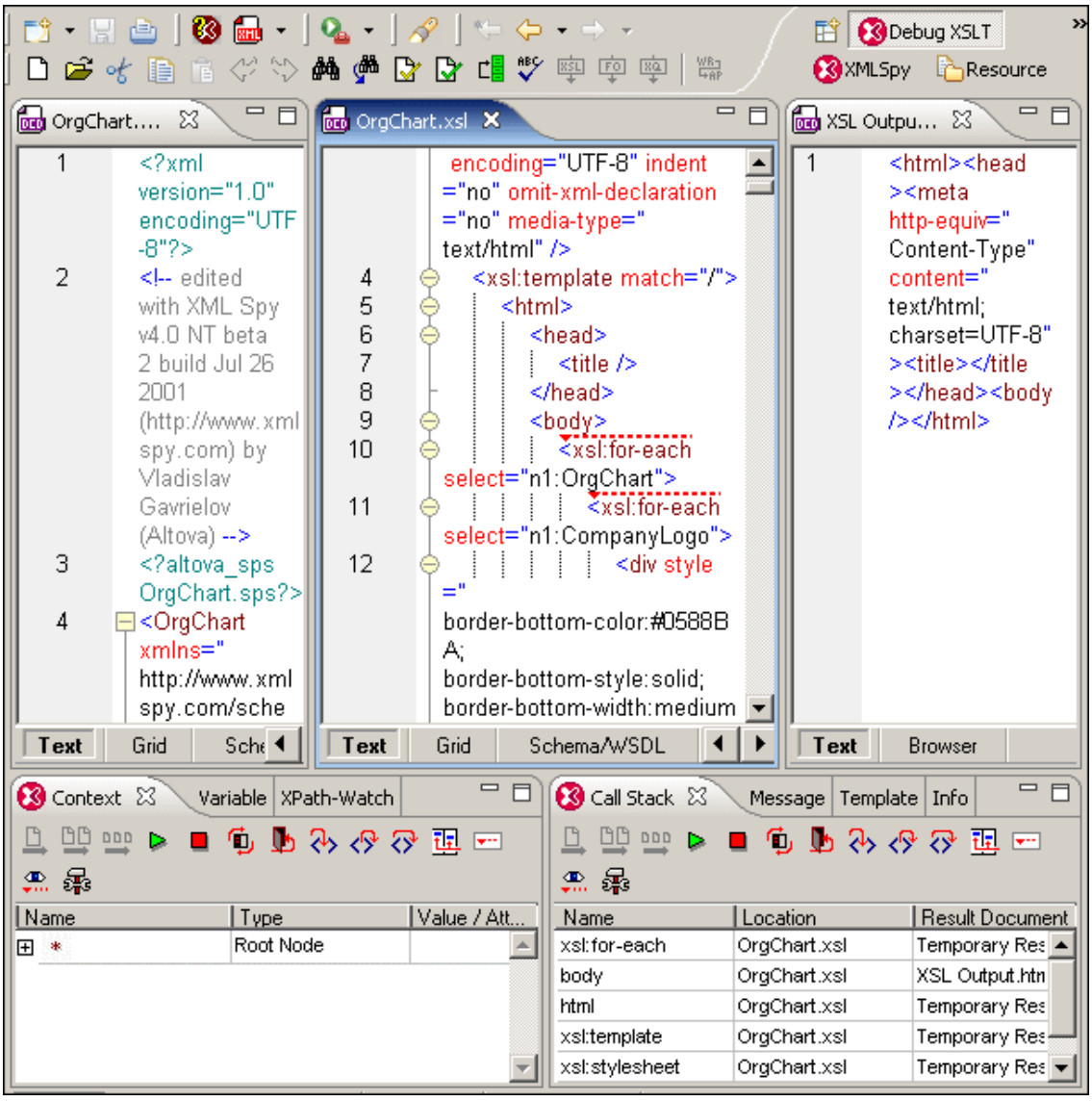
So debuggen Sie XSLT:

1. Öffnen Sie die Dateien **OrgChart.xml** und **Orgchart.xsl**.
2. Wählen Sie die Menüoption **XSL/XQuery | Debugger starten**.
3. Wählen Sie die gewünschte XML-Beispieldatei (klicken Sie auf die Schaltfläche "Fenster" und wählen Sie OrgChart.xml).
   Es erscheint eine Meldung, dass die Ansicht nun gewechselt wird. Bestätigen Sie mit OK.



Erstellen separater Dateifenster:

1. Klicken Sie auf das Register **XSL Output.html**, ziehen Sie es ganz nach rechts und lassen Sie die Maustaste los, wenn ein Pfeil erscheint. Dadurch wird das Register von der Registergruppe abgekoppelt.
   Beim Ziehen wird als Positionierungshilfe ein Fensterumriss angezeigt.
2. Entkoppeln Sie auch das Register **OrgChart.xsl** auf dieselbe Weise und positionieren Sie es zwischen die beiden Fenster.
   In der nachstehenden Grafik sehen Sie das neue Layout und den Debugging-Vorgang.

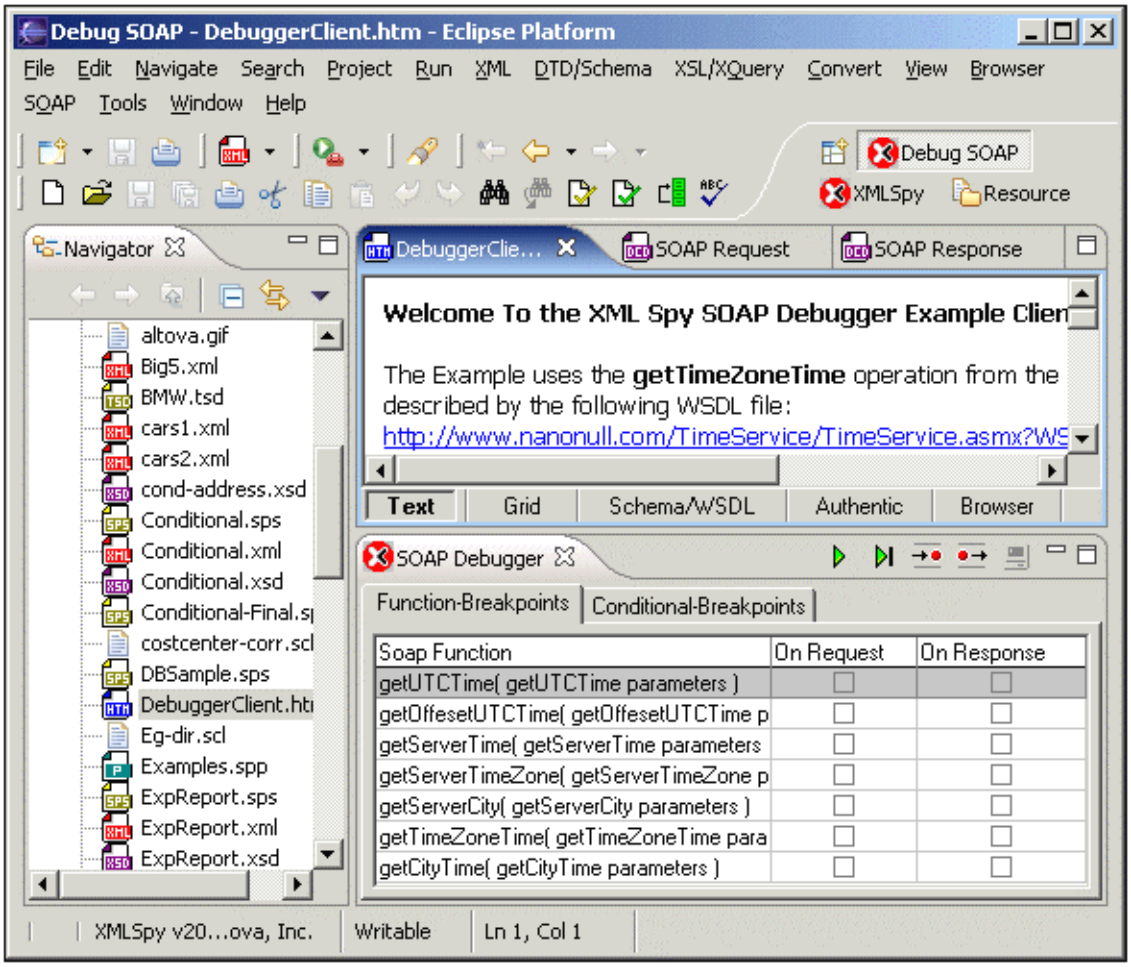


Wenn Sie die Debugger-Sitzung schließen, wird die XSL Output.html-Datei automatisch geschlossen und Sie wechseln in die XMLSpy-Perspektive zurück.

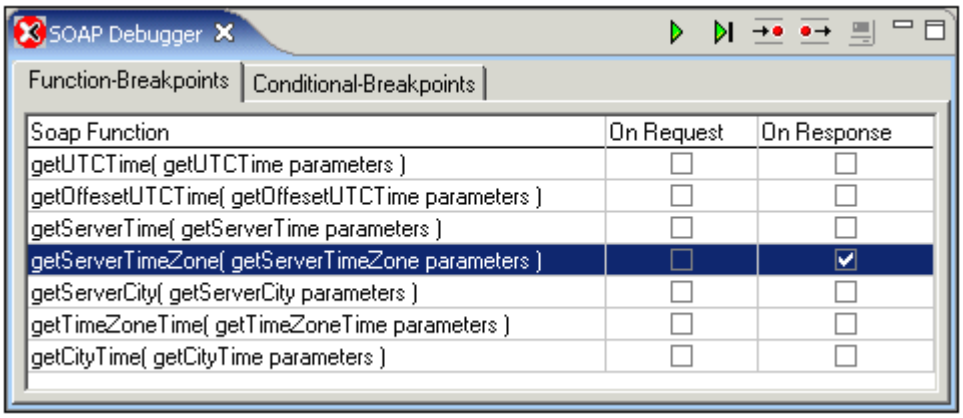
© 2007 Altova GmbH

10.6 Eclipse Plattform und SOAP Debugger

1. Öffnen Sie im Navigator die Datei **DebuggerClient.html**.
2. Wählen Sie die Menüoption **SOAP | SOAP Debugger-Sitzung**.
3. Wählen Sie im Ordner ..\Examples die Datei **TimeService.wsdl**.
4. Klicken Sie auf OK, um die Standard-Quell- und Zielports auszuwählen.

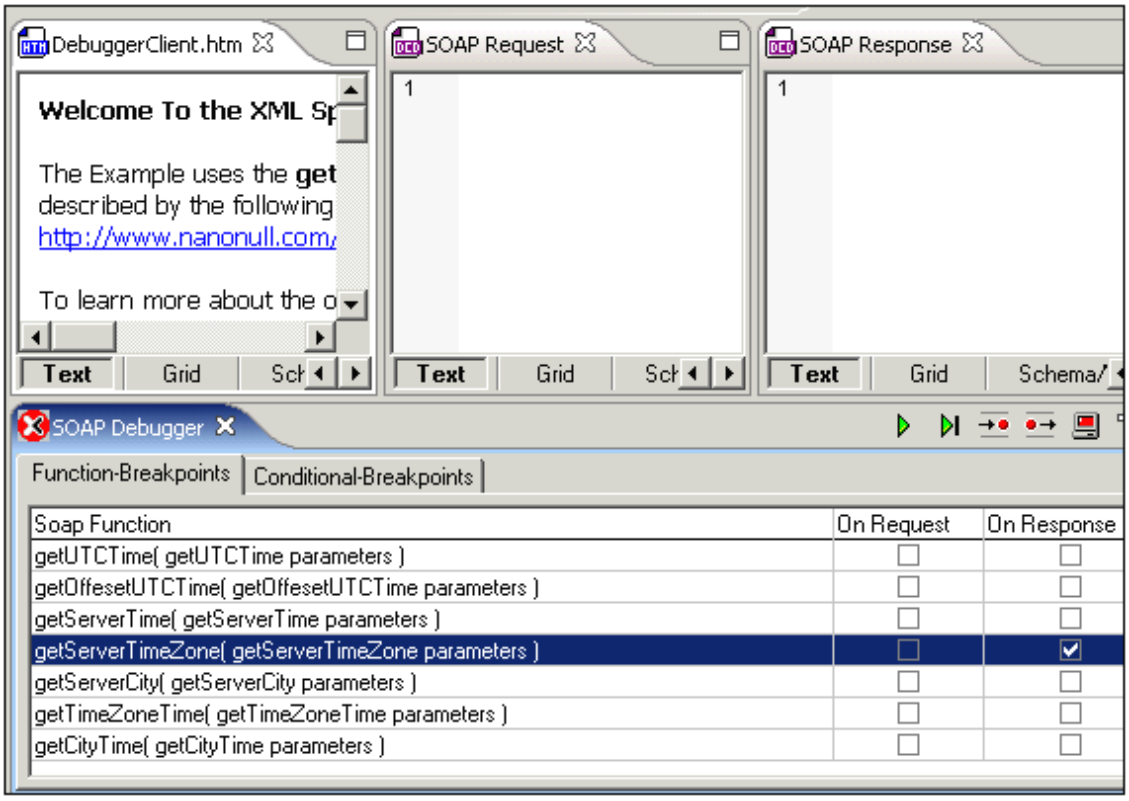


5. Setzen Sie auf dem SOAP Debugger-Registern einen Function Breakpoint (z.B. getServerTimeZone).



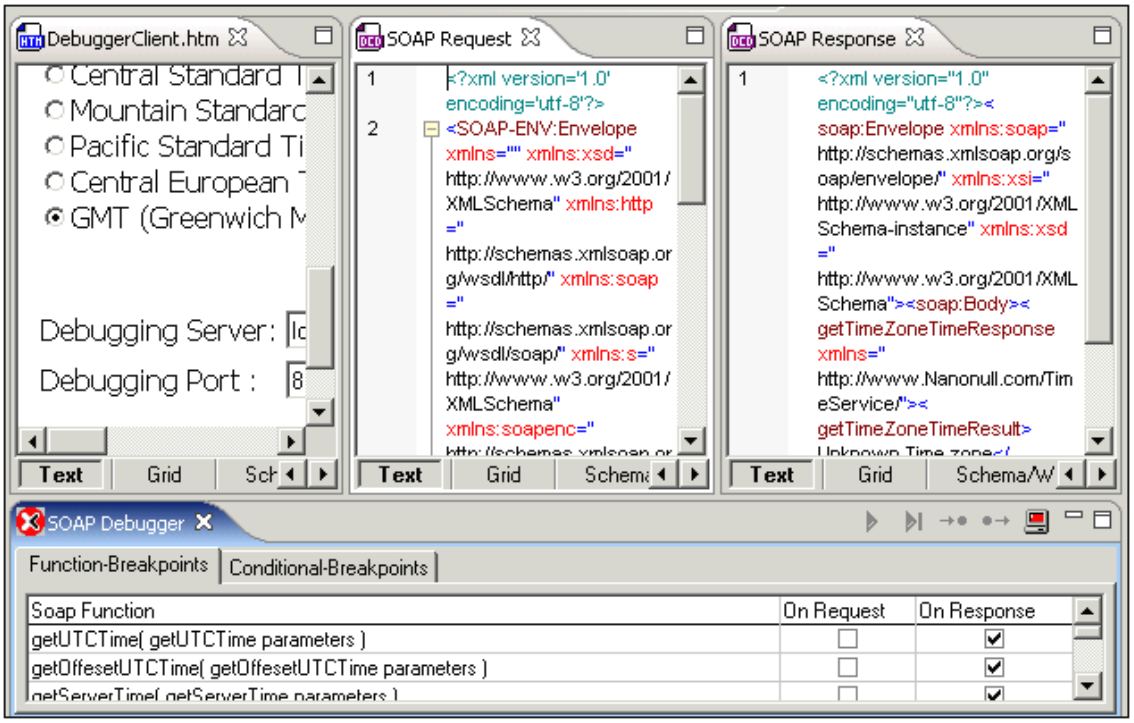
Erstellen separater Dateifenster:

1. Klicken Sie auf das Register "SOAP Response", ziehen Sie es ganz nach rechts und lassen Sie die Maustaste los, wenn ein Pfeil erscheint. Dadurch wird das Register von der Registergruppe abgekoppelt.
   Beim Ziehen wird als Positionierungshilfe ein Fensterumriss angezeigt.
2. Entkoppeln Sie auch das Register SOAP Request auf dieselbe Weise und positionieren Sie es zwischen die beiden Fenster



3. Wählen Sie die Menüoption **SOAP | Starten** oder klicken Sie auf das Symbol **Starten** im Debugger-Fenster.
4. Klicken Sie auf das Register **DebuggerClient.html,** scrollen Sie anschließend hinunter und klicken Sie auf die Schaltfläche **Turn on Debugging mode**.
5. Klicken Sie auf das Symbol "Beim nächsten Request anhalten" (oder auf das Optionsfeld GMT in der HTML-Datei) um den SOAP Request abzusenden.

 © 2007 Altova GmbH

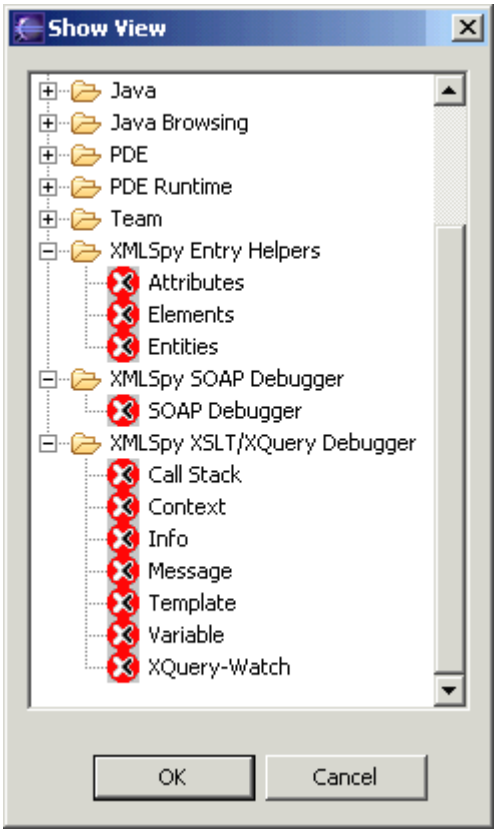


Wenn Sie die Debugger-Sitzung beenden, werden die Fenster SOAP Request und SOAP Response automatisch geschlossen und Sie wechseln in die XMLSpy -Perspektive zurück.

10.7 Eclipse-Ansichten und -Perspektiven

Die XMLSpy-Ansichten definieren die Haupt-Eingabehilfen sowie die Debugger-Fenster für die einzelnen Debugger.

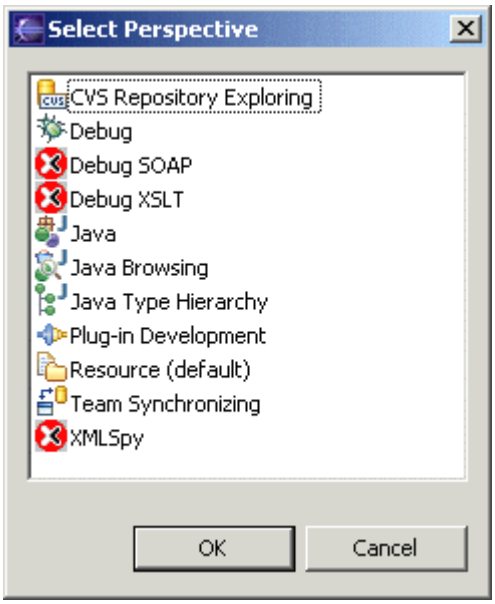
- Wählen Sie die Menüoption **Window | Show View | Other...,** um die aktuell verfügbaren Ansichten anzuzeigen.



Die XMLSpy-Perspektiven definieren das Anwendungs-Hauptfenster sowie die Debugger-Oberflächen.

- Wählen Sie die Menüoption **Window | Open Perspective | Other...,** um die aktuell verfügbaren Perspektiven anzuzeigen.

© 2007 Altova GmbH


Select Perspective
CVS Repository Exploring
Debug
Debug SOAP
Debug XSLT
Java
Java Browsing
Java Type Hierarchy
Plug-in Development
Resource (default)
Team Synchronizing
XMLSpy
OK
Cancel

11 XSLT und XQuery Debugger

Der XSLT/XQuery Debugger dient zum Testen und Debuggen von XSLT Stylesheets und XQuery-Dokumenten. Über die Oberfläche des XSLT/XQuery Debugger können Sie gleichzeitig eine Ansicht des XSLT/XQuery-Dokuments, des Ergebnisdokuments und des XML-Quelldokuments anzeigen. Sie können sich anschließend Schritt für Schritt durch das XML-XSLT/XQuery-Dokument bewegen. Bei jedem Schritt wird die entsprechende Ausgabe generiert. Wenn eine XML-Quelldatei angezeigt wird, wird die entsprechende Position in dieser Datei bei jedem Schritt markiert. Gleichzeitig zeigen Fenster der Benutzeroberfläche Informationen zur Behebung von Fehlern an.

Der XSLT/XQuery Debugger wird immer während einer **Debugger-Sitzung** geöffnet. Es gibt drei Arten von Debugger-Sitzungen:

- XSLT 1.0, wobei der integrierte Altova XSLT 1.0-Prozessor verwendet wird
- XSLT 2.0, wobei der integrierte Altova XSLT 2.0-Prozessor verwendet wird
- XQuery, wobei der integrierte Altova XQuery 1.0-Prozessor verwendet wird

Die Art der Debugger-Sitzung wird automatisch gewählt und hängt von der Art des Dokuments ab, von dem aus die Debugger-Sitzung gestartet wird (Dieses Dokument bezeichnen wir hier als das *aktive Dokument* oder die *aktive Datei.*). XSLT-Debugger-Sitzungen werden von XSLT-Dateien aus gestartet (Die Version hängt vom Wert des Attributs `version` des Elements `xsl:stylesheet` (oder `xsl:transform`) im XSLT Stylesheet ab (`"1.0"` für XSLT 1.0 und `"2.0"` für XSLT 2.0).). XQuery-Debugger-Sitzungen werden von XQuery-Dateien aus gestartet. Wenn die aktive Datei eine XML-Datei ist, hängt die Auswahl davon ab, ob Sie eine XSLT- oder eine XQuery-Datei auf die XML-Datei anwenden. In ersterem Fall hängt die Auswahl der Debugger-Art davon ab, ob es sich beim Stylesheet um ein XSLT 1.0- oder ein XSLT 2.0-Stylesheet handelt..

In der nachfolgenden Tabelle sehen Sie eine Zusammenfassung dieser Informationen.

Aktive Datei	Verknüpfte Datei	Debugger-Sitzung
XSLT 1.0	XML; (erforderlich)	XSLT 1.0 (mit dem integrierten Altova XSLT 1.0-Prozessor)
XSLT 2.0	XML; (erforderlich)	XSLT 2.0 (mit dem integrierten Altova XSLT 2.0-Prozessor)
XQuery	XML; (optional)	XQuery (mit dem integrierten Altova XQuery-Prozessor)
XML	XSLT 1.0, oder XSLT 2.0, oder XQuery; (erforderlich)	XSLT 1.0, XSLT 2.0 oder XQuery. XSLT 1.0 oder 2.0 je nach Wert des Attributs `version` des Elements `xsl:stylesheet` (oder `xsl:transform`) im XSLT Stylesheet.

Nähere Informationen zu den drei Altova-Prozessoren finden Sie in den folgenden Abschnitten:

- Altova XSLT 1.0-Prozessor
- Altova XSLT 2.0-Prozessor
- Altova XQuery 1.0-Prozessor

Automatisieren von XSLT- und XQuery-Aufgaben mit Altova XML 2007
Altova XML ist eine kostenlose Applikation, die den Altova XML-Validator sowie einen XLST 1.0-, XSLT 2.0- und XQuery 1.0-Prozessor enthält. Das Tool kann über die Befehlszeile, über eine COM-Schnittstelle, in Java-Programmen und in .NET-Applikationen zur Validierung von

 © 2007 Altova GmbH

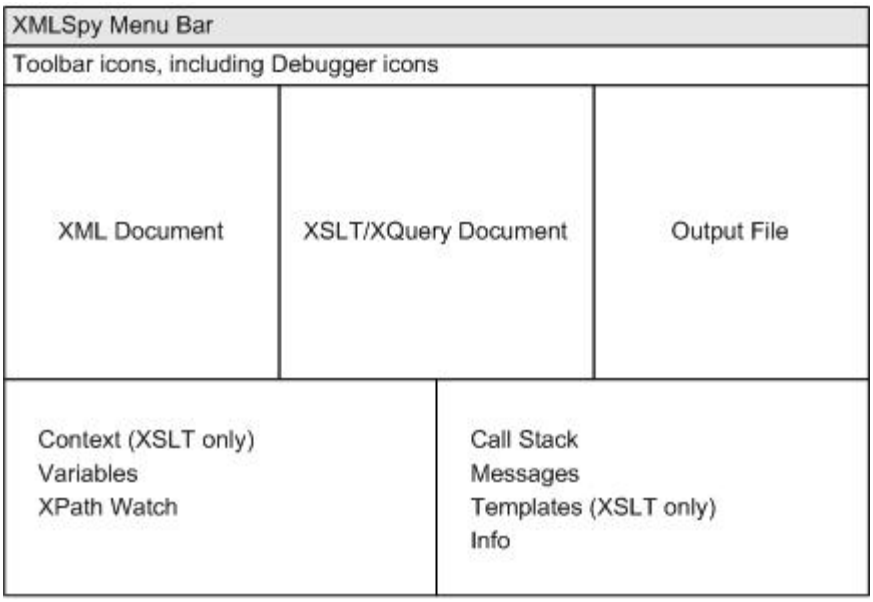
XML-Dokumenten, zum Transformieren von XML-Dokumenten unter Verwendung von XSLT 1.0 und 2.0 Stylesheets sowie zum Ausführen von XQuery-Dokumenten verwendet werden.

XSLT- und XQuery-Aufgaben können daher mittels Altova XMl automatisiert werden. So können Sie z.B. eine Batch-Datei erstellen, die Altova XML aufruft, um damit eine Reihe von XML-Dokumenten zu transformieren oder eine Reihe von XQuery-Dokumenten auszuführen. Nähere Informationen dazu finden Sie in der Altova XML Dokumentation.

11.1 Funktionsweise und Benutzeroberfläche

Die grundlegende Vorgangsweise beim Debuggen von XSLT- und XQuery-Dateien mit XSLT/XQuery Debugger ist die folgende:

- Öffnen Sie eine Debugger-Sitzung. Die Art der Sitzung (XSLT 1.0, XSLT 2.0 oder XQuery) wird anhand der aktiven Datei ausgewählt (*siehe* XSLT und XQuery Debugger). Der XSLT/XQuery Debugger kann nur in der Textansicht und der Enhanced Grid-Ansicht verwendet werden. Wenn die Ansicht des aktiven Dokuments beim Start der Debugger-Sitzung nicht die Textansicht oder die Enhanced Grid-Ansicht ist, werden Sie gefragt, ob in die Textansicht, also die Standardansicht des XSLT/XQuery Debugger gewechselt werden darf. Sie können diese Option auch permanent aktivieren.
- Gehen Sie das XSLT- bzw. XQuery-Dokument schrittweise durch. Wenn eine XML-Datei mit der Sitzung verknüpft ist, wird die jeweilige Stelle in der XML-Datei markiert. Gleichzeitig wird für jeden Schritt in der Ergebnisdatei eine Ausgabe generiert, d.h. das Ergebnisdokument wird Schritt für Schritt erzeugt. Auf diese Art können Sie analysieren, welche Auswirkung jede einzelne Anweisung in der XSLT- bzw. XQuery-Datei hat.



Alternativ zur Anzeige aller drei Dokumente (XML, XSLT/XQuery, Ergebnisdokument) (siehe Abb. oben), können Sie auch eine Ansicht mit zwei Dokumenten (XSLT/XQuery und Ergebnisdokument) oder eine Ansicht mit nur einem einzigen Dokument auswählen.

- Während eine Debugger-Sitzung geöffnet ist, werden auf der Benutzeroberfläche Informationsfenster mit Informationen über die verschiedenen Aspekte der Transformation/Ausführung angezeigt (Variablen, XPath Watch, Call Stack, Messages, Info, usw.).
- Sie können den Debugger während einer Debugger-Sitzung anhalten (dies ist nicht dasselbe wie die Beendung der Debugger-Sitzung), um Änderungen am Dokument vorzunehmen. Alle Editierfunktionen, die in Ihrer XMLSpy-Umgebung zur Verfügung stehen, können auch beim Editieren einer Datei während einer Debugger-Sitzung verwendet werden. Wenn der Debugger angehalten wird, bleibt die XSLT/XQuery Debugger-Oberfläche geöffnet und Sie haben Zugriff auf alle Informationen in den Debugger-Fenstern. Sie können nach dem Anhalten den Debugger in derselben Debugger-Sitzung erneut starten und das XSLT/XQuery-Dokument von Beginn an

 © 2007 Altova GmbH

erneut debuggen.

- Sie können in der XSLT-Datei Breakpoints setzen, um die Verarbeitung an bestimmten Punkten zu unterbrechen. Damit wird der Vorgang beschleunigt, da Sie nicht jede einzelne Anweisung im XSLT- bzw. XQuery-Dokument manuell durchgehen müssen.
- Sie können in der XSLT-Datei Tracepoints setzen. Bei Anweisungen in denen ein Tracepoint gesetzt wurde, wird der Wertt dieser Anweisung ausgegeben, sobald diese Anweisung erreicht wird.
- Beenden einer Debugger-Sitzung. Dadurch wird die XSLT/XQuery Debugger -Oberfläche geschlossen und Sie kehren zur vorher angezeigten XMLSpy Umgebung zurück. Die Informationen in den Informationsfenstern stehen nun nicht mehr zur Verfügung. Die Informationen zu den Breakpoints und Tracepoints bleiben jedoch in den jeweiligen Dateien erhalten, bis die Datei geschlossen wird. (Wenn Sie also eine weitere Debugger-Sitzung starten, bei der eine Datei mit Breakpoints verwendet wird, werden die Breakpoints in der neuen Debugger-Sitzung berücksichtigt.)

Bitte beachten Sie: Die Debugger-Symbolleiste und -Symbole werden automatisch angezeigt, wenn eine Debugger-Sitzung gestartet wird.

11.2 Befehle und Symbolleistenschaltflächen

Die Debugger-Befehle stehen im Menü **XSL/XQuery** und als Symbolleistenschaltflächen zur Verfügung. Die Debugger-Symbole erscheinen automatisch in der Symbolleiste, sobald eine Debugger-Sitzung geöffnet wird.

Debugger starten (Alt+F11)
Startet die Verarbeitung des XSLT/XQuery-Dokuments bzw. fährt damit fort, bis das Ende des Dokuments erreicht ist. Wurden Breakpoints gesetzt, so wird die Verarbeitung an diesen Stelle angehalten. Wenn der Debugger noch nicht gestartet wurde, starten Sie mit diesem Symbol die Sitzung, bis sie am ersten Node angehalten wird. Bei laufender Debugger-Sitzung wird das XSLT/XQuery-Dokument bis zu Ende verarbeitet oder bis der nächste Breakpoint erreicht wird. Wenn Tracepoints gesetzt wurden, wird der Wert der Anweisung für die der Tracepoint gesetzt wurde, im Fenster "Trace" angezeigt, wenn diese Anweisung erreicht wird.

Nur aktives Dokument ansehen
Maximiert das Fenster des gerade aktiven Dokuments auf der Debugger-Oberfläche.

XSLT/XQuery und Ausgabe ansehen
Zeigt das XSLT- und das Ausgabedokument an und blendet das XML-Dokument aus.

XML, XSLT/XQuery und Ausgabe anzeigen
Zeigt das XML-, XSLT/XQuery- und das Ausgabedokument an. Dies ist die Standardansicht, wenn ein XML-Dokument für die Debugger-Sitzung verknüpft wurde.

Debugger anhalten
Hält den Debugger an. Dies ist nicht dasselbe wie das Beenden der Debugger-Sitzung, in der der Debugger ausgeführt wird. Diese Funktion eignet sich dazu, um ein Dokument während der Debugger-Sitzung zu bearbeiten oder innerhalb derselben Debugger-Sitzung andere Dateien zu verwenden. Nachdem Sie den Debugger angehalten haben, müssen Sie ihn erneut starten, um am Anfang des XSLT/XQuery-Dokuments mit dem Debuggen zu beginnen.

Einsteigen (F11)
Geht die einzelnen Nodes und XPath-Ausdrücke im Stylesheet schrittweise durch. Dieser Befehl dient auch zum erneuten Starten des Debuggers, nachdem er angehalten wurde.

Überspringen (Strg+F11)

© 2007 Altova GmbH

Überspringt den aktuellen Node und geht zum nächsten Node auf derselben Ebene bzw. zum nächsten Node auf der nächsthöheren Ebene. Dieser Befehl dient auch zum erneuten Starten des Debuggers, nachdem er angehalten wurde.

Aussteigen (Umschalt+F11)
Steigt aus dem aktuellen Node aus und geht zum nächsten Geschwisternode des Parent-Node bzw. zum nächsthöheren Node oberhalb des Parent-Node.

Aktuellen Execution Node einblenden
Zeigt den aktuellen Execution Node im XSLT/XQuery-Dokument und den entsprechenden Kontext-Node im XML-Dokument an bzw. wählt diese aus. Dies ist nützlich, wenn Sie auf andere Register geklickt haben, die bestimmten Code im XSLT Stylesheet bzw. der XML-Datei anzeigen oder markieren, und zum Ausgangsort zurückkehren wollen.

Debugger neu starten
Löscht das Ausgabefenster und startet die Debugger-Sitzung mit den aktuell ausgewählten Dateien neu.

Breakpoint einfügen/löschen (F9)
Fügt einen Breakpoint an der aktuellen Cursorposition ein bzw. löscht diesen. Breakpoints innerhalb einer Zeile können für Nodes sowohl im XSLT/XQuery-Dokument als auch im XML-Dokument definiert werden. Sie legen fest, an welcher Stelle die Verarbeitung angehalten werden soll. Oberhalb des Nodes, an dem Sie einen Breakpoint gesetzt haben, erscheint eine strichlierte rote Linie. Für Abschlussnodes können keine Breakpoints gesetzt werden. Breakpoints in Attributen werden in XSLT-Dokumenten ignoriert. Der Befehl kann auch durch Rechtsklick auf die gewünschte Stelle aufgerufen werden.

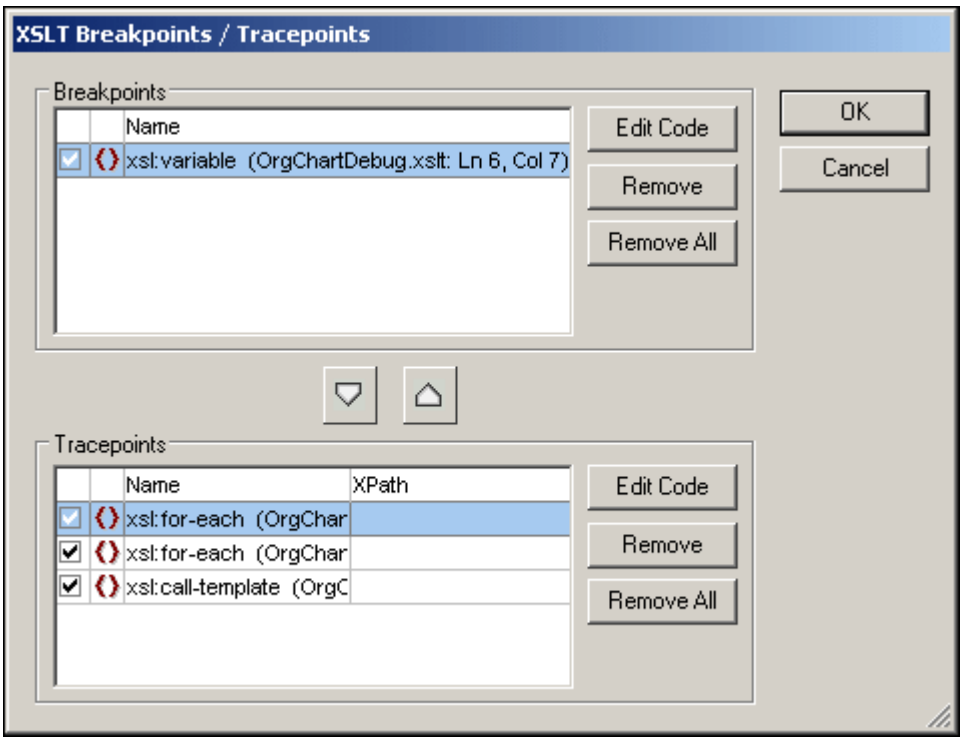
Tracepoint einfügen/löschen (Umschalt+F9)
Fügt einen Tracepoint an der aktuellen Cursorposition ein. In XSLT-Dokumenten können Tracepoints innerhalb einer Zeile gesetzt werden. Wenn beim Debuggen eine Anweisung mit einem Tracepoint erreicht wird, wird das Ergebnis dieser Anweisung im Fenster "Trace" ausgegeben. Oberhalb von Nodes, in denen ein Tracepoint gesetzt wurde, erscheint eine strichlierte blaue Linie. Tracepoints können nicht in schließenden Nodes gesetzt werden. Dieser Befehl steht auch zur Verfügung, wenn Sie mit der rechten Maustaste an die Stelle für den Tracepoint klicken.

Breakpoint aktivieren/deaktivieren (Strg+F9)
Dieser Befehl (in der Symbolleiste nicht verfügbar) schaltet bereits definierte Breakpoints ein oder aus. Ein Breakpoint, der in aktiviertem Zustand rot angezeigt wird, erscheint grau, wenn er deaktiviert ist. Der Debugger hält an deaktivierten Breakpoints nicht an. Um einen Breakpoint ein- bzw. auszuschalten, setzen Sie den Cursor in den jeweiligen Node-Namen und klicken Sie auf den Befehl "Breakpoint aktivieren/deaktivieren". Der Befehl kann auch durch Rechtsklick auf die gewünschte Stelle aufgerufen werden.

Tracepoint aktivieren/deaktivieren (Umschalt+Strg+F9)
Dieser Befehl (in der Symbolleiste nicht verfügbar) aktiviert oder deaktiviert bereits definierte Tracepoints. Ein Tracepoint, der in aktiviertem Zustand blau angezeigt wird, erscheint grau, wenn er deaktiviert ist. Bei deaktivierten Tracepoints wird im Fenster "Trace" keine Ausgabe angezeigt. Um einen Tracepoint zu deaktivieren bzw. zu aktivieren, setzen Sie den Cursor in den jeweiligen Node-Namen und klicken Sie auf den Befehl "Tracepoint aktivieren/deaktivieren". Der Befehl kann auch durch Rechtsklick auf die Stelle für den Tracepoint aufgerufen werden.

Debugger-Sitzung beenden
Beendet die Debugger-Sitzung und kehrt zur normalen XMLSpy-Ansicht zurück, die vor dem Start der Debugger-Sitzung aktiv war. Ob die Ausgabedokumente, die für die Debugger-Sitzung geöffnet wurden, weiterhin geöffnet bleiben, hängt von den Einstellungen ab, die Sie im Dialogfeld XSLT/XQuery Debugger-Einstellungen vorgenommen haben.

Dialogfeld "XSLT Breakpoints/Tracepoints"
Dieser Befehl ruft das Dialogfeld "XSLT/XQuery Breakpoints/Tracepoints" auf, in dem eine Liste aller in den Dateien der aktiven Debugger-Sitzung definierten Breakpoints/Tracepoints (einschließlich aller deaktivierten) angezeigt wird.



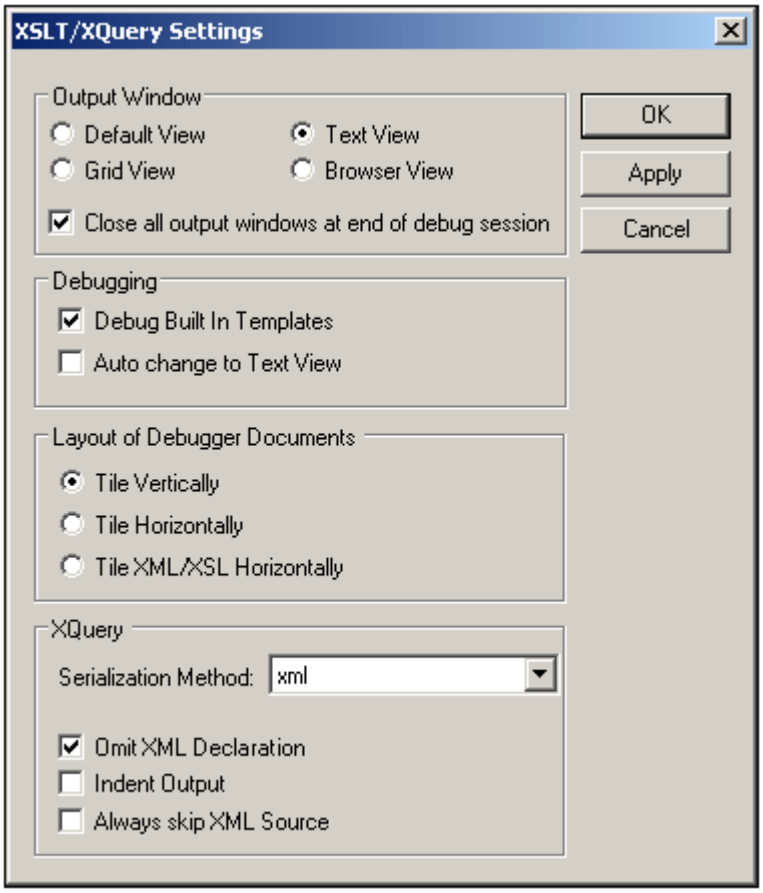
Die Kontrollkästchen zeigen an, ob ein Breakpoint/Tracepoint aktiviert (mit Häkchen versehen) oder deaktiviert ist. Sie können den markierten Breakpoint/Tracepoint durch Klicken auf die Schaltfläche **Löschen** entfernen. Bei Klicken auf die Schaltfläche **Alle löschen** werden alle

© 2007 Altova GmbH

Breakpoints bzw. Tracepoints gelöscht. Bei Auswahl der Schaltfläche **Code bearbeiten** wird der entsprechende Breakpoint/Tracepoint in der Datei direkt angezeigt.

11.3 Einstellungen

Im Dialogfeld XSLT/XQuery Debugger-Einstellungen können Sie Debugger- und Ausgabeoptionen einstellen, die für alle Debugger-Sitzungen gelten. Um das Einstellungsdialogfeld aufzurufen, wählen Sie den Befehl **XSL/XQuery | XSLT/XQuery-Einstellungen** oder klicken Sie auf das Symbol in der Symbolleiste. Die verschiedenen Einstellungen werden weiter unten beschrieben.



Ausgabefenster
Konfiguriert die Ansicht des Ausgabefensters (Standardansicht, Textansicht, Grid-Ansicht oder Browser-Ansicht). Die Standardansicht ist die Ansicht, die im Dialogfeld "Optionen" (**Extras | Optionen**) auf dem Register "Datei" für den Ausgabedateityp ausgewählt wurde. Für XSLT-Transformationen ist der Ausgabedateityp in der XSLT-Datei definiert. Für XQuery-Ausführungen ist der Ausgabedateityp durch die im Dialogfeld "XQuery-Einstellungen" ausgewählte "Serialization Method" festgelegt. (*siehe unten*).

Die Option "Alle Ausgabefenster nach Debuggen schließen" gibt Ihnen die Möglichkeit, die Ausgabefenster, die während der Debugger-Sitzung geöffnet waren, auch nach dem Schließen der Debugger-Sitzung geöffnet zu lassen.

Debugging
Bei Auswahl der Einstellung "Built-in Templates debuggen" **steigt** der Debugger, in built-in Templates **ein**, wann immer dies notwendig ist. Dies hat nichts mit der **Anzeige** von built-in Templates zu tun, wenn Sie auf dem Register "Templates" auf diese Art von Template-Eintrag klicken oder wenn der Callstack einen Node aus der Built-in Template-Datei anzeigt.

© 2007 Altova GmbH

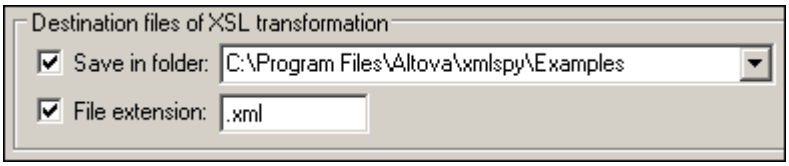
Der XSLT Debugger kann nur in der Text- oder der Grid-Ansicht verwendet werden. Bei Aktivierung der Option "Automatisch zu Textansicht wechseln" wird zum Debuggen automatisch in die Textansicht eines Dokuments gewechselt, wenn ein Dokument nicht in der Text- oder Grid-Ansicht angezeigt wird. (Der XQuery Debugger kann nur in der Textansicht verwendet werden.)

Layout von Debugger-Dokumenten
Die Debugger-Dokumente sind die Dokumente, die im Debugger geöffnet sind. Sie können auswählen, ob diese Dokumente vertikal oder horizontal bzw. mit der Option "XML/XSL horizontal anordnen" angeordnet werden sollen, wobei das Ergebnisdokument im letzteren Fall vertikal neben dem XML- und dem XSLT-Dokument angeordnet wird.

XQuery
Die "Serialization Method" bestimmt zwei Dinge: (i) das Dateiformat der generierten Ausgabedatei und (ii) die Regeln, die für das Schreiben der Ausgabe in die Ausgabedatei gelten. Zur Verfügung stehen die Optionen: HTML, Text, XHTML und XML. Je nach gewählter Option wird eine leere Datei des gewählten Dateityps erstellt, wenn Sie den Debugger innerhalb einer Debugger-Sitzung starten. Der Dateityp ist deshalb von Bedeutung, weil dadurch die aktuell aktiven XMLSpy-Funktionen für diesen Dateityp verfügbar werden (wie z.B. die Validierung für XML-Dateien).

Sie haben die Möglichkeit, die XML-Deklaration auszulassen und die Ausgabe einzurücken. Bei Aktivieren der Option "XML-Quelle immer überspringen" können Sie den optionalen Schritt zur Verknüpfung der XML-Datei immer überspringen, wenn Sie eine Debugger-Sitzung von einer XQuery-Datei aus starten.

Einstellung der Auswahl eines anderen Ausgabeformats
Im Dialogfeld "Projekteigenschaften" können Sie für XSLT-Transformationen Verknüpfungen für einen Ordner erstellen. Eine dieser Optionen ist die Auswahl des Zielordners für die XSLT-Transformation, für den Sie eine Dateierweiterung wählen können.



Der hier ausgewählte Dateityp bestimmt den Dateityp des Ausgabeformats für XML- oder XSLT-Dateien, die zu einem Projekt gehören, für das XSLT-Transformationseigenschaften definiert wurden.

11.4 Starten einer Debugger-Sitzung

Am einfachsten lässt sich eine Debugger-Sitzung von einer XSLT-, XQuery- oder XML-Datei aus starten. Wenn der aktiven Datei bereits die erforderliche Datei (siehe Tabelle der verknüpften Dateien) zugewiesen wurde, wird die Debugger-Sitzung sofort gestartet. Andernfalls werden Sie aufgefordert, die erforderliche Datei zuzuweisen. Da XQuery-Dateien eine Verknüpfung mit einer XML-Datei weder benötigen noch enthalten, können Sie einstellen, ob Sie bei jedem Start einer XQuery Debugger-Sitzung (von einer XQuery-Datei aus) aufgefordert werden sollen, eine optionale XML-Dateiverknüpfung anzugeben, oder nicht.

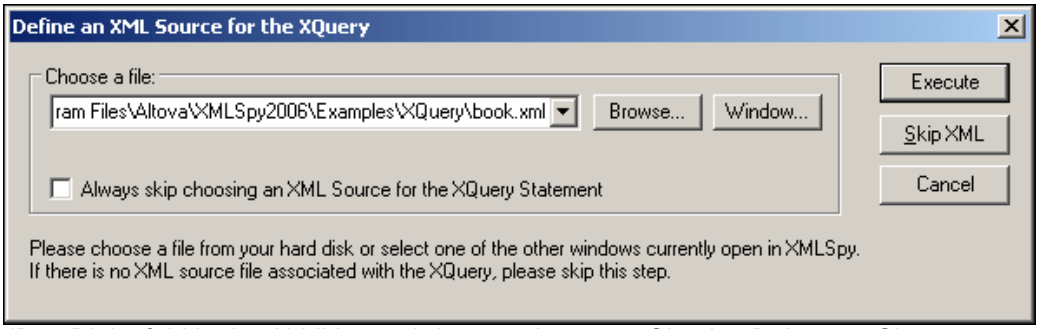
Vordefinierte Verknüpfungen
Vordefinierte Verknüpfungen sind nur für XSLT Debugger-Sitzungen relevant und beziehen sich auf Fälle, in denen die Dateiverknüpfung bereits in der aktiven Datei vorhanden ist. Um in einer XML- oder XSLT-Datei eine Zuweisung vorzunehmen, gehen Sie folgendermaßen vor:

- In XML-Dateien: Öffnen Sie die Datei, wählen Sie **XSL/XQuery | XSL zuweisen** und wählen Sie die XSLT-Datei aus.
- In XSLT-Dateien: Öffnen Sie die Datei, wählen Sie den Befehl **XSL/XQuery | XML-Beispieldatei zuweisen...** und wählen Sie die XML-Datei aus.

Wenn Sie den Befehl **XSL/XQuery | Debugger starten** wählen, wird die Debugger-Sitzung direkt gestartet, d.h. ohne dass Sie aufgefordert werden, die aktive Datei mit einer Datei zu verknüpfen.

Direkte Zuweisung
Falls die aktive Datei keine vordefinierte Verknüpfung enthält, werden Sie aufgefordert, eine anzugeben. Wenn Sie den Befehl **XSL/XQuery | Debugger starten** wählen, geschieht Folgendes:

- bei XML-Dateien: Sie werden aufgefordert, eine XSLT- oder XQuery-Datei auszuwählen.
- bei XSLT-Dateien: Sie werden aufgefordert, eine XML-Datei auszuwählen.
- bei XQuery-Dateien: Sie haben die Option, eine XML-Datei auszuwählen. Diese Option kann übersprungen werden.



(Das Dialogfeld in der Abbildung wird angezeigt, wenn Sie eine Debugger-Sitzung von einer XQuery-Datei aus starten.)

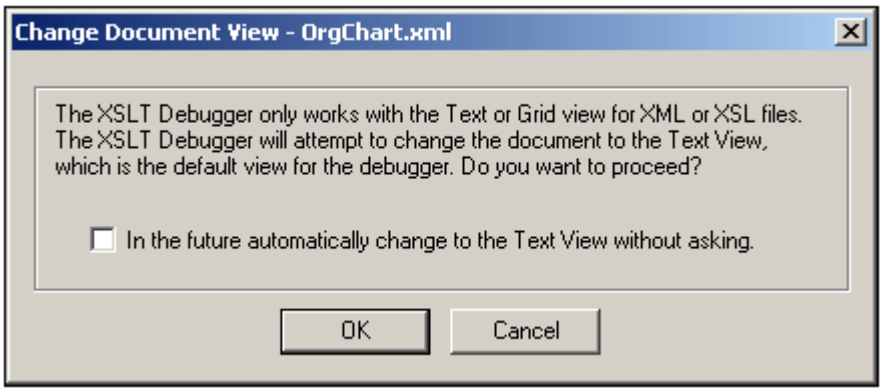
Nach Auswahl der erforderlichen Verknüpfungsdatei bzw. Auslassen der optionalen Verknüpfung wird die Debugger-Sitzung gestartet.

Alternative Methode zur Dateizuordnung
Im Dialogfeld "Projekteigenschaften" können Sie Verknüpfungen vordefinieren. Klicken Sie auf **Projekt | Projekteigenschaften** und nehmen Sie die gewünschten Dateizuweisungen vor,

 © 2007 Altova GmbH

indem Sie das Kontrollkästchen **XSL-Vorlage** / **XML-Vorlage** aktivieren.

Debugger-Ansicht
Der XSLT/XQuery Debugger funktioniert nur in der Text- und der Grid-Ansicht. Wenn die jeweilige XML- oder XSLT-Datei in einer anderen Ansicht geöffnet ist, bzw. wenn eine SPS-Datei mit einer XML-Datei verknüpft ist, erscheint das folgende Dialogfeld, wenn Sie eine Debugger-Sitzung unter Verwendung einer dieser Dateien starten.

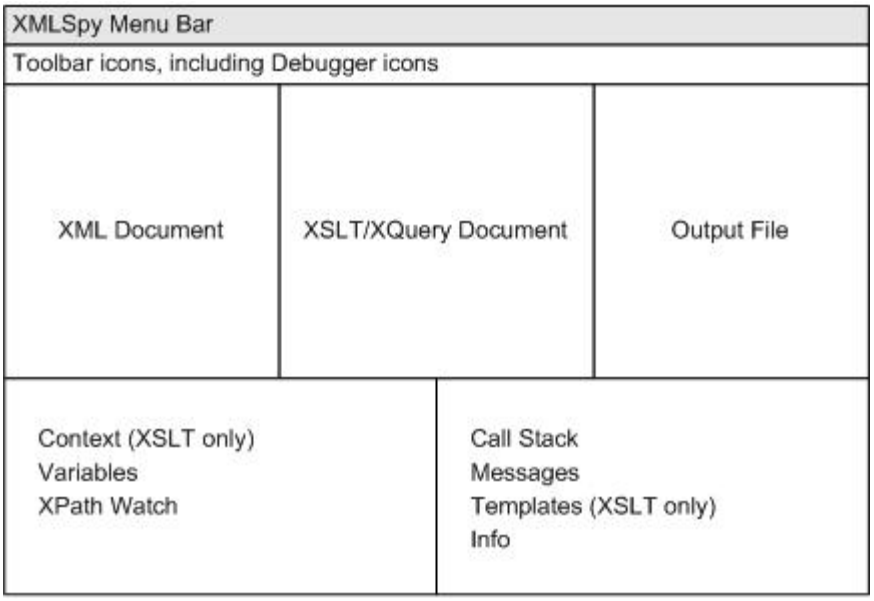


Wenn Sie auf OK klicken, wird das Dokument in der Textansicht geöffnet. Beachten Sie, dass XQuery-Dateien immer in der Textansicht angezeigt werden.

11.5 Informationsfenster

Informationsfenster, die während der Debugger-Sitzung im XSLT/XQuery Debugger -Anwendungsfenster geöffnet werden, enthalten Informationen über verschiedene Debugging-spezifische Aspekte der XSLT-Transformation bzw. der XQuery-Ausführung.

In XSLT-Debugger-Sitzungen stehen acht Informationsfenster zur Verfügung, in XQuery-Debugger-Sitzungen fünf. Diese Fenster sind standardmäßig in zwei Gruppen unterteilt und sind am unteren Rand der XSLT/XQuery Debugger-Oberfläche (*siehe Abbildung unten*) angeordnet. Die Fenster und die darin angezeigten Informationen werden in diesem Abschnitt näher beschrieben.



Standard-Layout der XSLT/XQuery Debugger-Oberfläche.

In der ersten Informationsfenstergruppe werden die folgenden Fenster in Form von Registerkarten in einem einzigen Fenster angezeigt:

- Kontext (nur bei XSLT Debugger-Sitzungen)
- Variablen
- XPath-Watch

In der zweiten Informationsfenstergruppe werden die folgenden Fenster in Form von Registerkarten in einem einzigen Fenster angezeigt:

- Call Stack
- Messages
- Templates (nur bei XSLT Debugger-Sitzungen)
- Info
- Trace

Im Standard-Layout gibt es zwei Fenstergruppen, jeweils mit Registern für die verschiedenen Fenster darin. Ein Register darin ist jeweils aktiv. Wenn Sie z.B. Informationen über Variablen anzeigen möchten, klicken Sie in der ersten Informationsfenstergruppe auf das Register "Variablen". Daraufhin wird das Variablenfenster angezeigt, während die Informationsfenster "Kontext" und "XPath-Watch" ausgeblendet werden. Beachten Sie, dass Sie auf einigen Registern die darin angezeigten Informationen auch zum Navigieren verwenden können: Wenn Sie auf einen Eintrag klicken, wird das jeweilige Element/Attribut in der XML-, XSLT- oder XQuery-Datei angezeigt. Nähere Informationen dazu finden Sie in der Dokumentation zu den jeweiligen Informationsfenstern (Kontext, Call Stack, Templates).

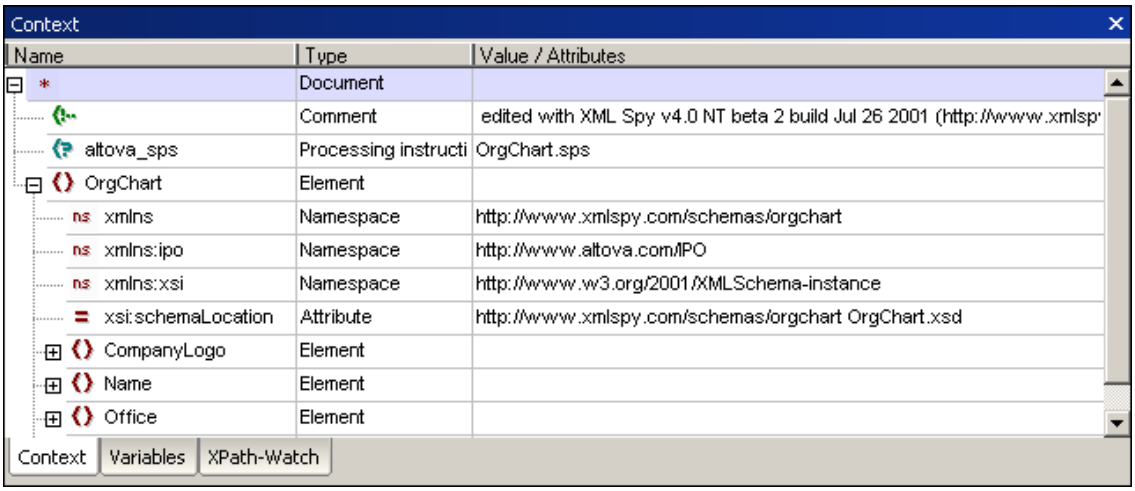
© 2007 Altova GmbH

Durch Ziehen der Umrandung der beiden Informationsfenstergruppen können Sie deren Größe anpassen. Einzelne Fenster können aus der Gruppe herausgezogen werden, indem Sie auf den Registernamen klicken und das Fenster aus der Gruppe herausziehen. Um ein Fenster zur Gruppe hinzuzufügen, ziehen Sie seine Titelleiste auf die Titelleiste der Gruppe. Beachten Sie, dass es keine Schaltfläche zum Zurücksetzen auf das Standard-Layout gibt.

11.5.1 Kontext

Das Fenster "Kontext" steht nur in XSLT Debugger-Sitzungen nicht aber in XQuery Debugger-Sitzungen zur Verfügung.

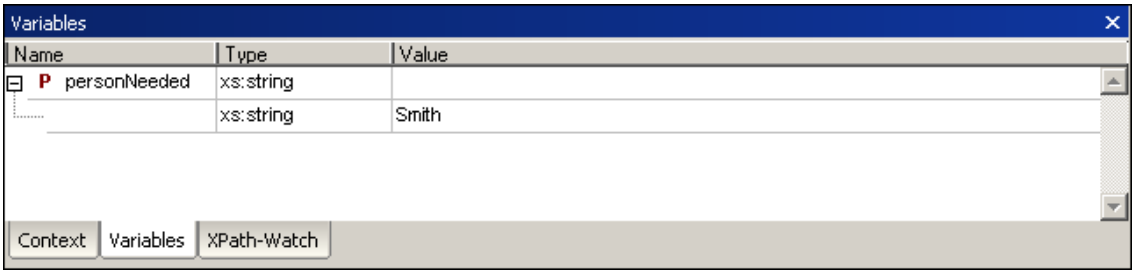
Während der Verarbeitung eines XSLT Stylesheet befindet sich der Prozessor-Kontext immer in einer Template, die mit einer Sequenz (von Nodes oder atomaren Werten) übereinstimmt. Im Kontextfenster wird der aktuelle Verarbeitungskontext angezeigt. Dabei kann es sich um eine Node-Sequenz, einen einzelnen Node oder einen atomaren Wert (wie z.B. einen String) handeln. Je nach Kontext-Eintrag, wird/werden der Wert bzw. das/die Attribut(e) angezeigt. Wenn es sich beim Kontext-Eintrag z.B. um ein Element handelt, werden die Attribute des Elements angezeigt. Wenn es sich beim Kontext-Eintrag um ein Attribut oder einen Text-Node handelt, wird der Wert des Node angezeigt.



Wenn Sie auf einen Eintrag im Kontextfenster klicken, wird diese Stelle im XML-Dokument angezeigt, Wenn das XML-Dokument derzeit auf der Benutzeroberfläche nicht angezeigt wird, wird ein Fenster dafür geöffnet.

11.5.2 Variablen

Das Fenster "Variablen" steht in XSLT und XQuery Debugger-Sitzungen zur Verfügung. Es werden darin die Variablen und Parameter und deren Werte angezeigt, die im XSLT/XQuery-Dokument verwendet werden, wenn Sie im Geltungsbereich liegen.

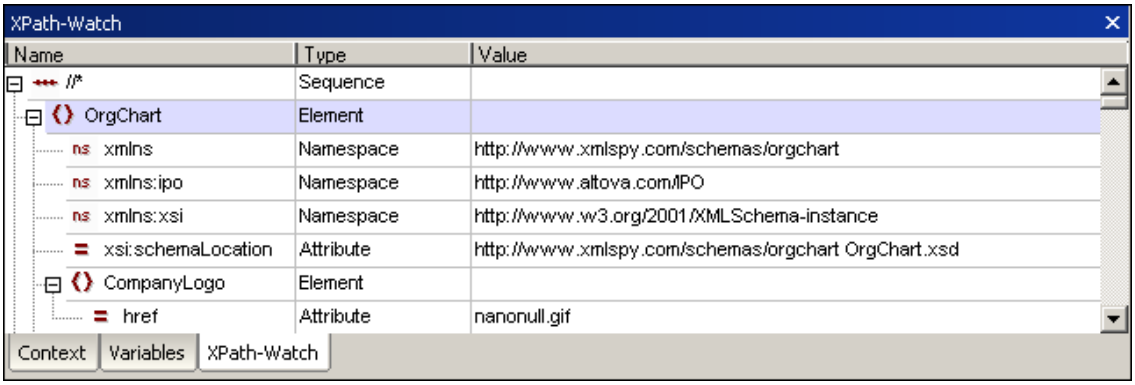


Parameter werden durch den Buchstaben **P** gekennzeichnet, globale Variablen (Variablen auf der obersten Ebene eines Stylesheet) durch ein **G** und lokale Variablen (innerhalb einer XSLT-Template deklarierte) durch ein **L**. Die Art des Werts von Variablen und Parameters wird im Feld "Value" ebenfalls durch Symbole gekennzeichnet. Es wird zwischen den folgenden Arten unterschieden: Node Set, Node Fragment, String, Number und Boolean.

11.5.3 XPath-Watch

Das Fenster "XPath-Watch" steht in XSLT- und XQuery-Debugger-Sitzungen zur Verfügung.

Sie können über dieses Fenster XPath-Ausdrücke eingeben, die Sie in einem oder mehreren Kontexten überprüfen möchten. Während Sie das XSLT-Dokument Schritt für Schritt überprüfen, wird der XPath-Ausdruck im aktuellen Kontext evaluiert und das Ergebnis wird in der Spalte "Value" angezeigt.



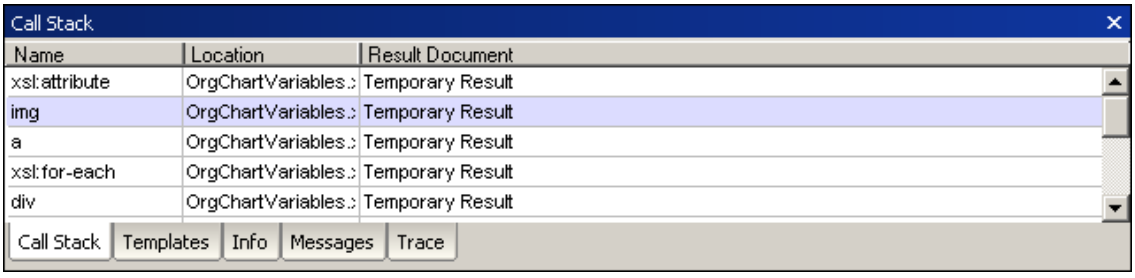
Um einen XPath-Ausdruck einzugeben, doppelklicken Sie in das Textfeld unterhalb der Spalte "Name" und geben Sie den XPath ein. Verwenden Sie Ausdrücke, die gemäß der XPath-Version, die der XSLT-Version des XSLT Stylesheet entspricht (XPath 1.0 für XSLT 1.0 und XPath 2.0 für XSLT 2.0), korrekt sind.

Bitte beachten Sie: Wenn in der XML- bzw. XSLT-Datei Namespaces verwendet wurden, müssen Sie in Ihren XPath-Ausdrücken die richtigen Namespace-Präfixe verwenden.

11.5.4 Call Stack

Das Fenster "Call Stack" wird in XSLT- und XQuery Debugger-Sitzungen angezeigt.

Im Fenster "Call Stack" wird eine Liste aller zuvor verarbeiteten XSLT Templates und -Anweisungen angezeigt, wobei die aktuelle Template/Anweisung als erste in der Liste angezeigt wird.



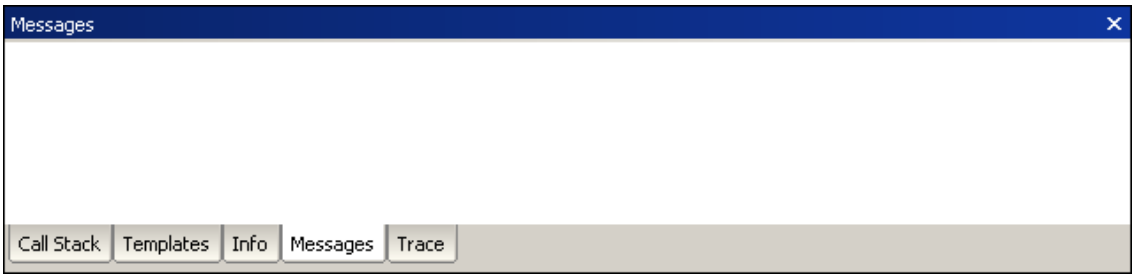
 © 2007 Altova GmbH

Wenn Sie in diesem Fenster auf einen Eintrag klicken, wird die jeweilige XSLT-Template/Anweisung im XSLT-Dokumentenfenster angezeigt. Wenn Sie auf eine Template/Anweisung klicken, die eine Built-in Template referenziert, wird die jeweilige built-in Template in einem separaten Fenster angezeigt, in dem alle Built-in Templates zu sehen sind.

11.5.5 Messages

Das Fenster "Messages" wird in XSLT- und XQuery-Debugger-Sitzungen angezeigt.

XSLT 1.0 und XSLT 2.0
In XSLT Debugger-Sitzungen werden auf dem Register "Messages" Fehlermeldungen, die `xsl:message`-Anweisung(en) oder andere Fehlermeldungen, die beim Debuggen auftreten können, angezeigt.

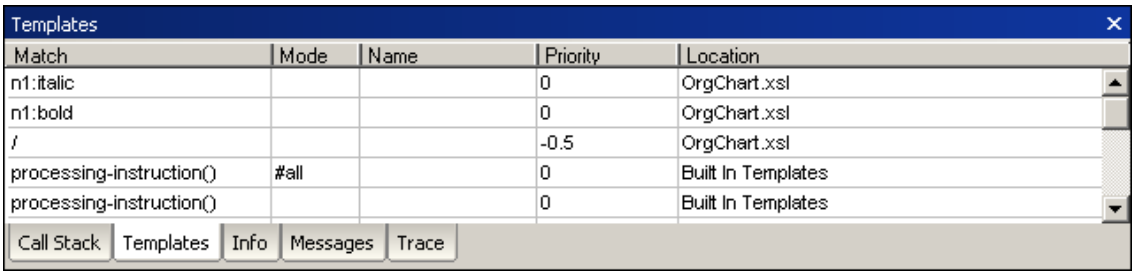


XQuery
In XQuery-Debugger-Sitzungen werden im Fenster "Messages" Fehlermeldungen angezeigt.

11.5.6 Templates

Das Fenster "Templates" (*siehe Abbildung unten*) ist nur in XSLT Debugger-Sitzungen verfügbar, nicht aber in XQuery Debugger-Sitzungen.

Im Fenster "Templates" werden die verschiedenen in XSLT Stylesheets verwendeten Templates angezeigt - einschließlich Built-in Templates und Named Templates. Matched Templates werden nach den Nodes aufgelistet, mit denen diese übereinstimmen. Named Templates werden nach Namen aufgelistet. Für beide Template-Arten werden Mode (Modus), Priority (Priorität) und Location (Pfad) angezeigt.



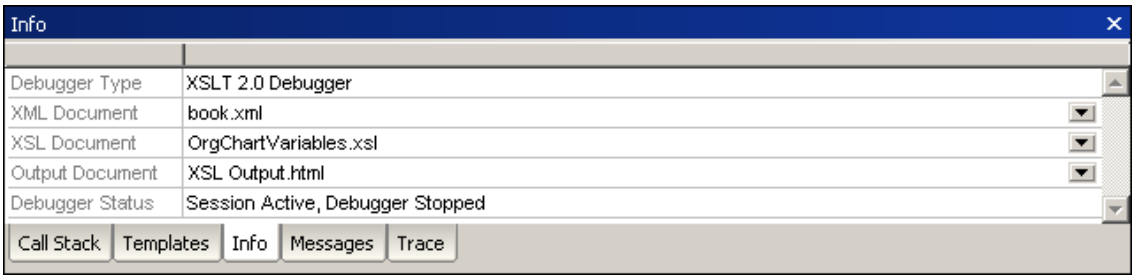
In oben gezeigten Screenshot sehen Sie zwei Templates im XSLT Stylesheet: Eine Template, die mit dem Dokument Node `/` übereinstimmt und eine Template mit dem Namen `tokenize`. Alle anderen Templates sind Built-in Templates (die im Feld "Location" durch keinen Eintrag gekennzeichnet sind).

Wenn Sie auf einen Eintrag in diesem Fenster klicken, wird die Template im XSLT-Dokumentenfenster markiert. Wenn Sie auf eine Built-in Template klicken, wird die

Template in einem separaten Fenster, in dem alle Built-in Templates zu sehen sind, markiert.

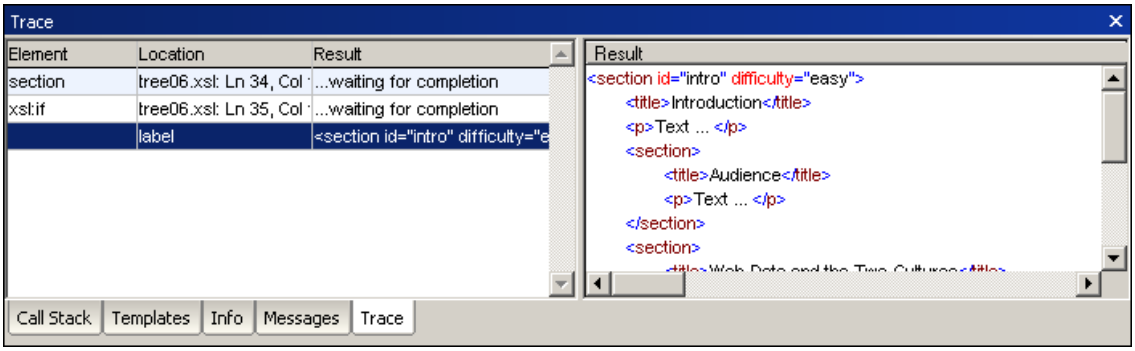
11.5.7 Info

Das Fenster "Info" (*siehe Abbildung unten*) ist in XSLT- und XQuery Debugger-Sitzungen verfügbar und enthält Informationen über die aktuelle Debugger-Sitzung wie u.a. Informationen zur Version des verwendeten Debuggers, die Namen des Quell- und Zieldokuments und den Status des Debuggers.



11.5.8 Trace

Das Fenster "Trace" wird in XSLT- und XQuery-Debugger-Sitzungen angezeigt.



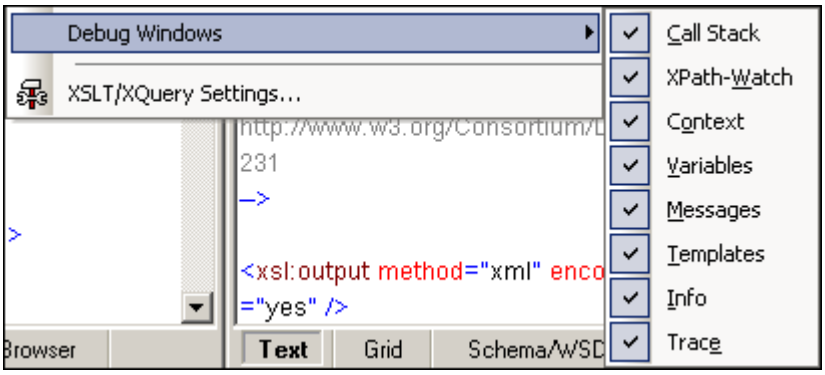
Das Fenster "Trace" enthält das Element, für das der Tracepoint gesetzt wurde, seine Position im XSLT Stylesheet und das Ergebnis, das bei Ausführung des Elements generiert wird. Klicken Sie auf eine Zeile auf der linken Seite des Fensters um das vollständige Ergebnis auf der rechten Seite anzuzeigen.

11.5.9 Anordnen der Informationsfenster

Die Informationsfenster können an verschiedenen Stellen des XSLT/XQuery Debugger -Anwendungsfensters angeordnet werden. Fenster können angedockt werden, freischwebend angeordnet oder als mehrere Fenster in einem Fenster angezeigt werden. Zum Anordnen der Fenster stehen Ihnen die folgenden Möglichkeiten zur Verfügung.

Menü
Platzieren Sie den Cursor im Menü "XSL/XQuery" über den Eintrag "Debug-Fenster", um eine Liste von Info-Fenstern anzuzeigen. Durch Klicken auf den entsprechenden Eintrag können Sie bestimmte Fenster ein- oder ausblenden.

© 2007 Altova GmbH



Damit blenden Sie das Fenster ein- bzw. aus.

Kontextmenü
Das Kontextmenü wird durch Rechtsklick auf ein Register oder die Titelleiste eines Fensters aufgerufen.

Klicken Sie auf die gewünschte Option, um das Fenster an- oder abzudocken oder auszublenden.

Drag-and-Drop
Sie können ein Fenster an seinem Register oder der Titelleiste an die gewünschte Stelle ziehen.

Außerdem können Sie das Fenster beim Ziehen mit Hilfe von Steuerelementen zur Platzierung an einem anderen Fenster oder im Anwendungsfenster andocken. Diese Ziehpunkte werden angezeigt, wenn Sie ein Fenster ziehen:

- Wenn Sie ein Fenster über ein anderes Fenster ziehen, erscheint ein kreisförmiges Steuerelement, das aus fünf Sektoren besteht (*siehe Screenshot unten*). Wenn Sie die Maustaste über einem dieser Sektoren loslassen, wird das Fenster, das Sie gerade gezogen haben, am entsprechenden Bereich des **Zielfensters** angedockt. Mit Hilfe der vier Pfeilsektoren wird das Fenster an der entsprechenden Seite des Zielfensters angedockt. Mit Hilfe der Schaltfläche in der Mitte wird das Fenster als Register des Zielfensters angedockt. Sie können ein Fenster auch als Register in einem anderen Fenster andocken, indem Sie es in die Registerleiste ziehen.

- Beim Ziehen eines Fensters erscheint ein Steuerelement bestehend aus vier Pfeilen. Jeder Pfeil entspricht einer Seite der **Debugger-Oberfläche**. Wenn Sie die Maustaste beim Ziehen eines Fensters über einem dieser Pfeile loslassen, wird das Fenster an einer Seite der Debugger-Oberfläche angedockt.

Außerdem können Sie zwischen der an- und der abgedockten Position eines Fensters wechseln, indem Sie auf die Titelleiste des Fensters doppelklicken.

© 2007 Altova GmbH

11.6 Breakpoints

Der XSLT/XQuery Debugger gestattet Ihnen, Breakpoints in XSLT-, XQuery- und XML-Dokumenten zu definieren. Breakpoints werden als strichlierte rote Linie angezeigt (*siehe Abb. unten*).

Bitte beachten Sie: Sie können für dieselbe Anweisung einen Tracepoint und einen Breakpoint setzen. Angezeigt wird dies durch eine strichlierte blaurote Linie (*siehe Screenshot*).

```
<!-- 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>
```

Wenn Sie den Debugger in einer Debugger-Sitzung starten, wird der Debugger an jedem Breakpoint angehalten. Auf diese Art können Sie bestimmte Bereiche, auf die Sie Ihre Aufmerksamkeit im XSLT-, XQuery- und/oder XML-Dokument richten möchten, für das Debuggen kennzeichnen. Sie können beliebig viele Breakpoints definieren.

Bitte beachten Sie: Die für ein Dokument gesetzten Breakpoints bleiben erhalten, bis das Dokument geschlossen wird.

Breakpoints in XML-Dokumenten
Sie können an jedem Node in einem XML-Dokument einen Breakpoint setzen. Die Verarbeitung wird am Beginn des Node angehalten.

Breakpoints in XSLT-Dokumenten
Sie können an den folgenden Stellen in einem XSLT-Dokument Breakpoints setzen:

- am Anfang von Templates und Template Instructions (z.B. `xsl:for-each`).
- an einem XPath-Ausdruck (XPath 1.0 oder XPath 2.0).
- an jedem Node in einem literal erstellten XML-Fragment. Die Verarbeitung wird am Beginn des Node unterbrochen.

Breakpoints in XQuery-Dokumenten
Sie können an den folgenden Stellen in einem XQuery-Dokument Breakpoints setzen:

- am Beginn von XQuery-Anweisungen.
- in einem XQuery-Ausdruck.
- an jedem Node in einem hartcodierten XML-Fragment. Die Verarbeitung wird am Beginn des Node unterbrochen.

Einfügen/Löschen von Breakpoints
So fügen Sie einen Breakpoint ein:

1. Setzen Sie den Cursor an die Stelle im Dokument, an der der Breakpoint gesetzt werden soll (*siehe oben*). In XSLT Debugger-Sitzungen können Sie sowohl in der Text- als auch in der Grid-Ansicht Breakpoints setzen. XQuery-Debugger-Sitzungen stehen nur in der Textansicht zur Verfügung.

2. Wählen Sie eine der folgenden Methoden:

- Wählen Sie **XSL/XQuery | Breakpoint einfügen/löschen**.
- Drücken Sie die Taste **F9**.
- Klicken Sie auf die rechte Maustaste und wählen Sie **Breakpoint einfügen/löschen**.

So löschen Sie einen Breakpoint:

1. Setzen Sie den Cursor an die Stelle im Dokument, die den Breakpoint enthält.
2. Wählen Sie eine der folgenden Methoden:

- Wählen Sie **XSL/XQuery | Breakpoint einfügen/löschen**.
- Drücken Sie die Taste **F9**.
- Klicken Sie auf die rechte Maustaste und wählen Sie **Breakpoint einfügen/löschen**.

Alternativ dazu können Sie einen Breakpoint auch über das Dialogfeld "Breakpoints" löschen:

1. Wählen Sie die Menüoption **XSL/XQuery | Breakpoints...**.
2. Klicken Sie im Dialogfeld auf den Breakpoint und klicken Sie auf **Löschen**.

Über die Schaltfläche **Alle löschen** löschen Sie alle Breakpoints aus dem Dialogfeld (und allen XSLT Stylesheets).

Deaktivieren/Aktivieren von Breakpoints:
Sie können Breakpoints, die übersprungen werden sollen, auch deaktivieren ohne den Breakpoint zu löschen. Sie können diese Breakpoints später auch wieder aktivieren.

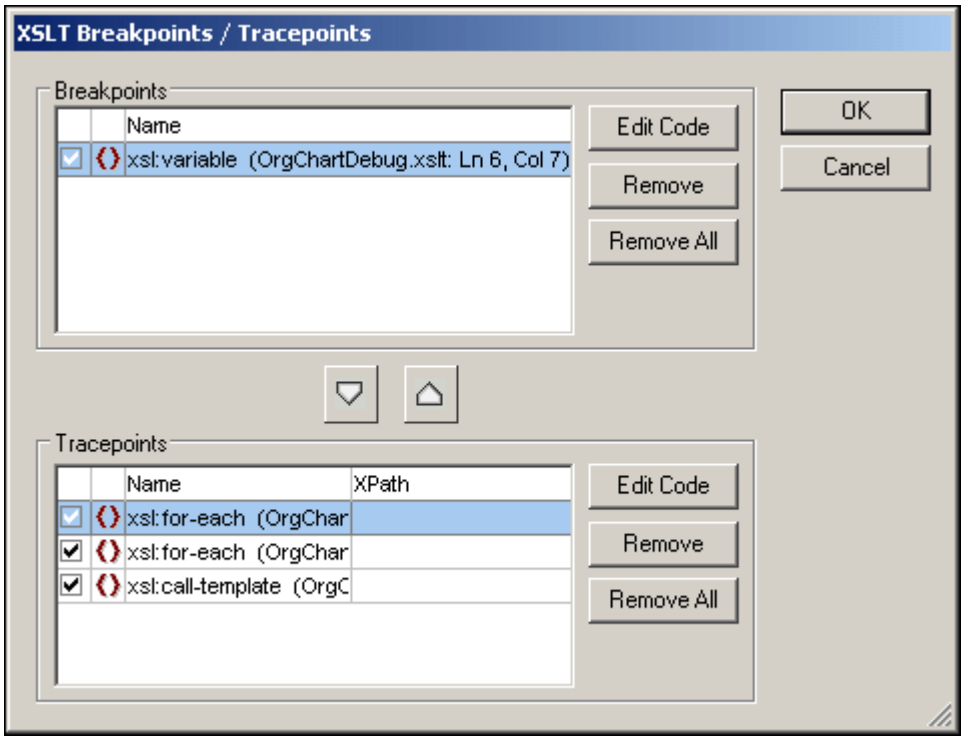
So deaktivieren Sie einen Breakpoint:

1. Setzen Sie den Cursor in den Node bzw. den Ausdruck, der den Breakpoint enthält.
2. Wählen Sie den Befehl **XSL/XQuery | Breakpoint aktivieren/deaktivieren** (Befehl kann auch mit Hilfe der Tastenkombination **Strg+F9** aufgerufen werden). Der Breakpoint ändert sich von rot in grau, was bedeutet, dass er deaktiviert wurde.

Alternativ dazu können Sie Breakpoints auch über das Dialogfeld "Breakpoints" deaktivieren. Gehen Sie dabei vor wie folgt:

1. Wählen Sie den Menübefehl **XSL/XQuery | Breakpoints/Tracepoints...**. Daraufhin wird das Dialogfeld "XSLT Breakpoints" geöffnet, in dem die aktuell definierten Breakpoints in allen XML-Quell- und XSLT Stylesheet-Dokumenten angezeigt werden.

© 2007 Altova GmbH



2. Entfernen Sie das Häkchen bei den Breakpoints, die Sie deaktivieren möchten und klicken Sie auf OK. Die entsprechenden Breakpoints werden nun statt rot grau angezeigt.

So aktivieren Sie einen Breakpoint:

1. Setzen Sie den Cursor in den Node bzw. Ausdruck, der den Breakpoint enthält.
2. Wählen Sie den Befehl **XSL/XQuery | Breakpoint ein/aus** (oder drücken Sie **Strg+F9**). Die Farbe des Breakpoints ändert sich von grau auf rot, d.h. der Breakpoint wurde aktiviert.

Suchen eines bestimmten Breakpoint
So suchen Sie nach einem bestimmten Breakpoint:

1. Wählen Sie die Menüoption **XSL/XQuery | Breakpoints/Tracepoints...**.Daraufhin wird das Dialogfeld "XSLT Breakpoints/Tracepoints" angezeigt.
2. Klicken Sie auf den gewünschten Breakpoint in der Liste der Breakpoints.
3. Klicken Sie auf die Schaltfläche **Code bearbeiten**. Das Dialogfeld "Breakpoints" wird geschlossen und der Textcursor wird in der Textansicht direkt vor den Breakpoint gesetzt. In der Enhanced Grid-Ansicht wird die Tabellenzelle mit dem Breakpoint rot markiert.

Fortsetzen der Debugger-Sitzung nach dem Breakpoint
So fahren Sie nach dem Breakpoint mit der Debugger-Sitzung fort:

- Wählen Sie den Befehl **XSL/XQuery | Einsteigen** oder **XSL/XQuery | Debugger starten**.

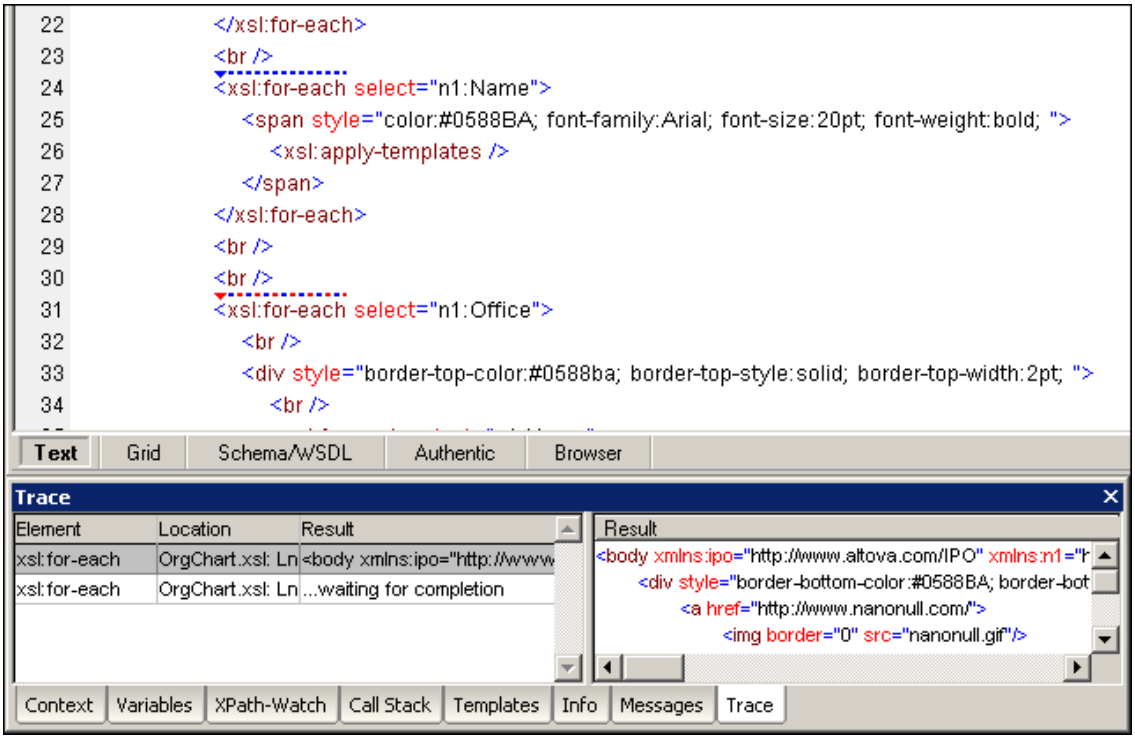
11.7 Tracepoints

Mit Hilfe des XSLT/XQuery Debugger können Sie Tracepoints in XSLT-Dokumenten definieren.

Mit Hilfe von Tracepoints können Sie Inhalt, der von einer Anweisung generiert wird, auffinden oder das Ergebnis eines XPath-Ausdrucks an dem Punkt anzeigen, an dem der Tracepoint gesetzt wurde, ohne das XSLT Stylesheet z.B. durch Verwendung des `xsl:message` Elements zur Ausgabe von Debugging-Meldungen bearbeiten zu müssen.

Tracepoints werden durch eine strichlierte blaue Linie im Code des XSLT Stylesheet angezeigt (*siehe Screenshot unten*).

Bitte beachten Sie: Sie können innerhalb derselben Anweisung einen Tracepoint und einen Breakpoint setzen. Dies wird in Form einer blaurot gestrichelten Linie dargestellt (*siehe Screenshot*).



Der Debugger gibt die Ergebnisse der einzelnen Anweisungen, in denen ein Tracepoint gesetzt wurde, aus. Dieses Ergebnis wird im Fenster "Trace" angezeigt. Sie können in einem XSLT Stylesheet beliebig viele Tracepoints setzen.

Bitte beachten Sie: Die Tracepoints bleiben in einem Dokument erhalten, bis es geschlossen wird.

Tracepoints in XSLT-Dokumenten
Sie können in einem XSLT Stylesheets in XSLT-Anweisungen und literalen Ergebnissen Tracepoints setzen.

Tracepoints in XML- und XQuery-Dokumenten
Sie können in XML- und XQuery-Dokumenten Tracepoints setzen, jedoch haben diese hier keine Auswirkung.

Einfügen/Löschen von Tracepoints

So fügen Sie einen Tracepoint ein:

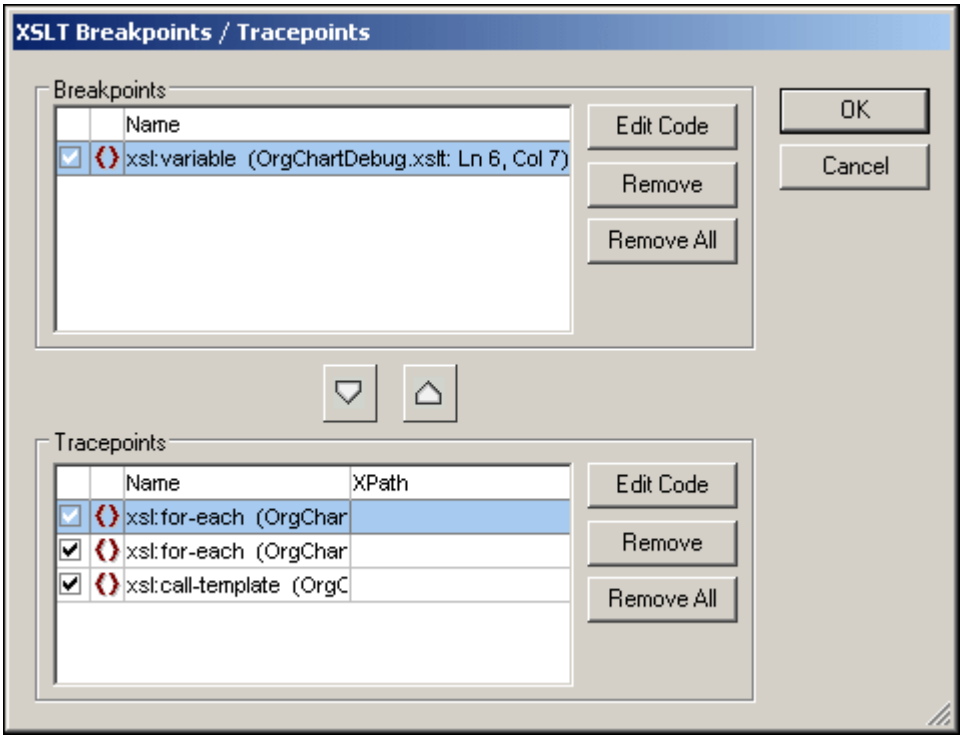
1. Platzieren Sie den Cursor an dem Punkt im XSLT-Dokument, an dem Sie den Tracepoint einfügen möchten. Sie können Tracepoints sowohl in der Text- als auch in der Grid-Ansicht setzen.
2. Wählen Sie eine der folgenden Methoden:
   - Wählen Sie **XSL/XQuery | Tracepoint einfügen/löschen**.
   - Drücken Sie **Umschalt+F9**.
   - Rechtsklicken Sie und wählen Sie den Befehl **Tracepoint einfügen/löschen.**

So löschen Sie einen Tracepoint:

1. Platzieren Sie den Cursor an die Stelle im XSLT-Dokument, die den Tracepoint enthält.
2. Wählen Sie eine der folgenden Methoden:
   - Wählen Sie **XSL/XQuery | Tracepoint einfügen/löschen**.
   - Drücken Sie **Umschalt+F9**.
   - Rechtsklicken Sie und wählen Sie den Befehl **Tracepoint einfügen/löschen**.

Alternativ dazu können Sie Tracepoints auch über das Dialogfeld "XSLT Breakpoints / Tracepoints" löschen:

1. Wählen Sie die Menüoption **XSL/XQuery | Breakpoints/Tracepoints...**.
2. Klicken Sie auf den Tracepoint im Dialogfeld und klicken Sie auf **Entfernen**.



Mit Hilfe der Schaltfläche **Alle löschen** löschen Sie die Tracepoints aus dem Dialogfeld (und allen XSLT Stylesheets).

Definieren eines XPath als Tracepoint
Sie können einen XPath als Tracepoint definieren. Wenn Sie einen XPath als Tracepoint definieren, wird im Fenster "Trace" anstelle des Inhalts, der durch die Anweisung, für die der Tracepoint gesetzt wurde, generiert wird, das Ergebnis der XPath-Auswertung angezeigt. Der XPath wird relativ zum Kontext-Node an der Stelle, an der der Tracepoint gesetzt wurde, ausgewertet.

So definieren Sie einen XPath als Tracepoint:

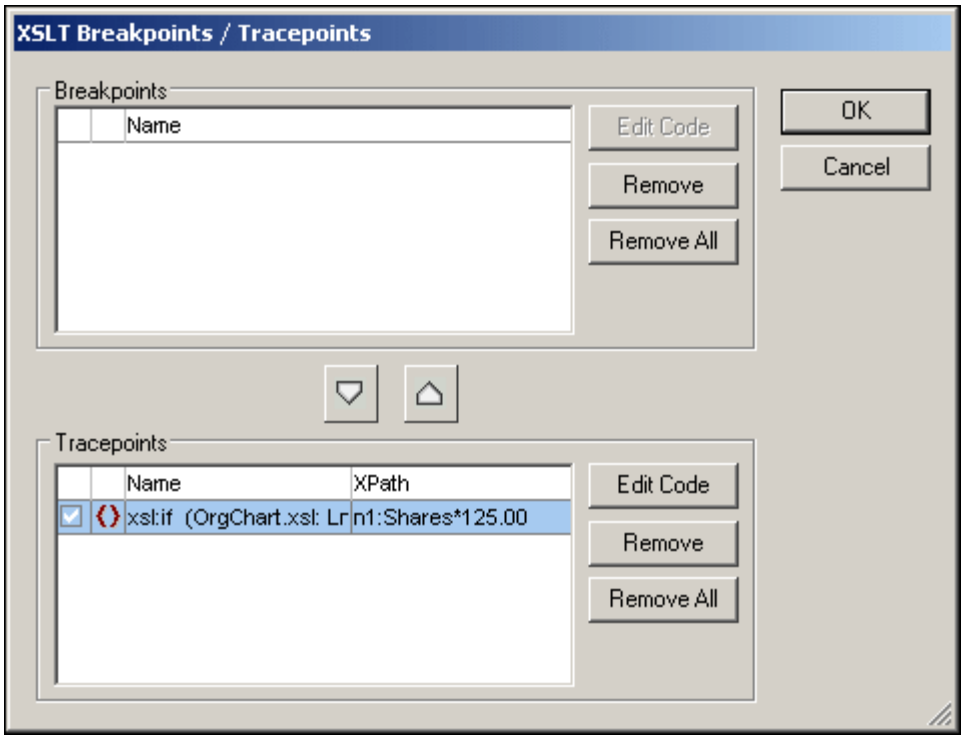
1. Wählen Sie die Menüoption **XSL/XQuery | Breakpoints/Tracepoints...**. Daraufhin wird das Dialogfeld "XSLT Breakpoints / Tracepoints" geöffnet, in dem alle derzeit definierten Tracepoints in allen offenen XSLT Stylesheets angezeigt werden.
2. Geben Sie den XPath in die Spalte **XPath** in der Zeile für den entsprechenden Tracepoint ein.

Beispiel:

Im folgenden Beispiel wird die Datei `OrgChart.xsl`, verwendet, die Sie im Ordner XMLSpy Examples finden.
Der Tracepoint wird so gesetzt, dass der Kontext Node "Person" ist. Das Element "Person" enthält ein Element "Shares". Wir möchten die Anzahl der Aktien (Shares), die jede Person hat, multipliziert mit 125 (dem Wert einer einzelnen Aktie) anzeigen.

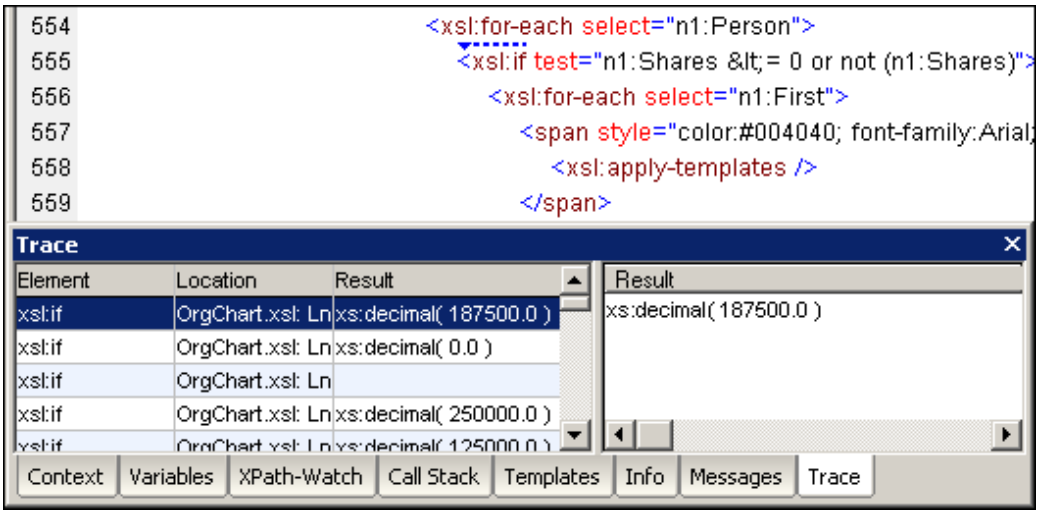
Gehen Sie folgendermaßen vor:

1. Öffnen Sie die Datei `OrgChart.xsl`.
2. Setzen Sie in Zeile 555 einen Tracepoint.
3. Öffnen Sie das Dialogfeld "XSLT Breakpoints / Tracepoints" und geben Sie den XPath `n1:Shares*125.00` für diesen Tracepoint ein.



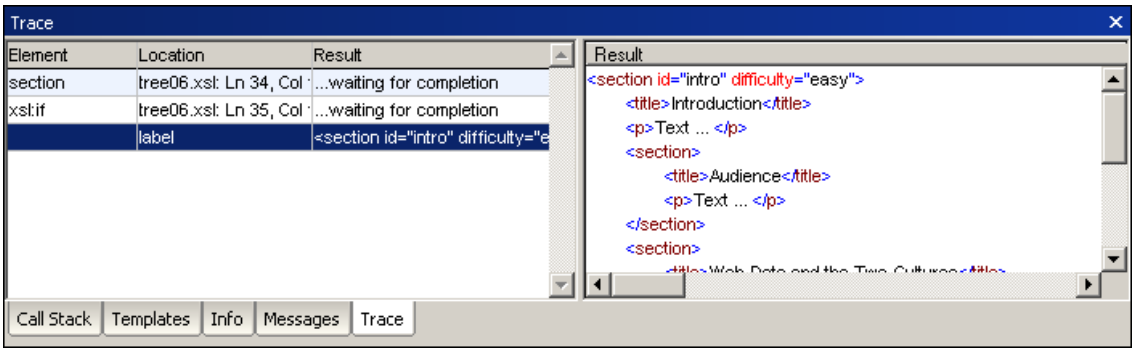
4. Starten Sie den Debugger. Das Ergebnis des XPath, den Sie als Tracepoint eingegeben haben, wird im Fenster "Trace" angezeigt.

© 2007 Altova GmbH



Das Fenster "Trace"
Wählen Sie den Befehl **XSL/XQuery** | **Debugger starten,** um den Debugger zu starten. Das Ergebnis der Anweisungen, für die Tracepoints gesetzt wurden, wird im Fenster Trace (*siehe Screenshot*) angezeigt. Klicken Sie auf eine Zeile im Fenster "Trace", um das vollständige Ergebnis der Anweisung auf der rechten Seite des Dialogfelds (*siehe Screenshot*) anzuzeigen.

Bitte beachten Sie: Die Ergebnisse werden erst dann im Fenster "Trace" angezeigt, wenn diese Trace-Anweisung fertig ausgeführt wurde.



Aktivieren/Deaktivieren von Tracepoints:
Sie können Tracepoints, nachdem Sie sie eingefügt haben, deaktivieren, wenn diese übersprungen aber nicht gelöscht werden sollen. Bei Bedarf können Sie diese jederzeit wieder aktivieren.

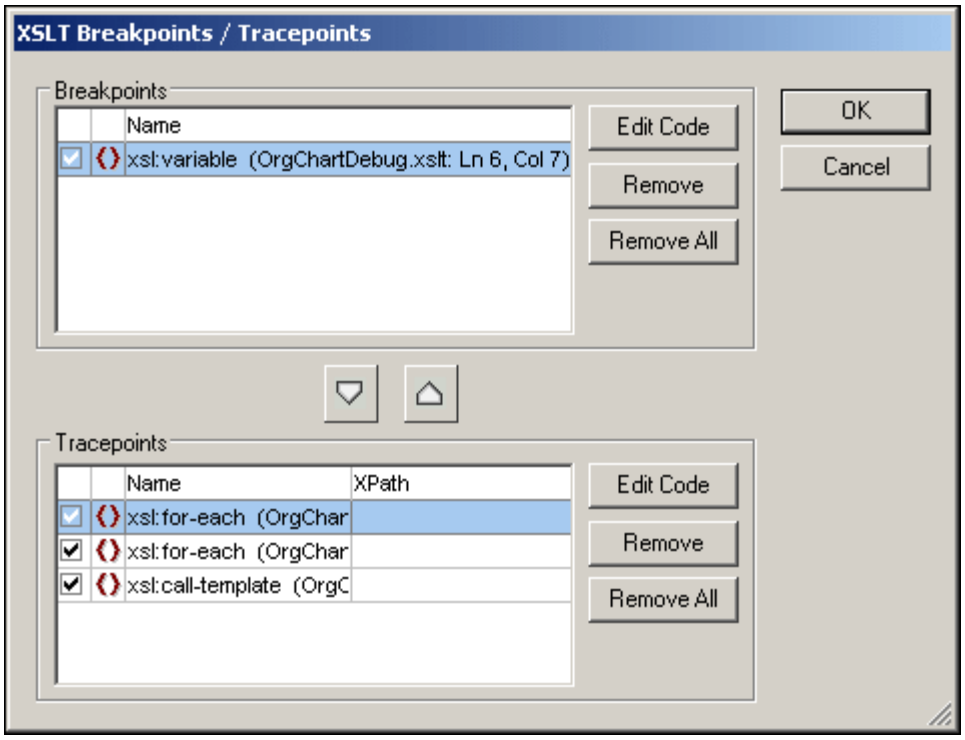
So deaktivieren Sie einen Tracepoint:

1. Platzieren Sie den Cursor an die Stelle im XSLT Stylesheet, die den Tracepoint enthält.
2. Wählen Sie **XSL/XQuery | Tracepoint aktivieren/deaktivieren** (oder drücken Sie **Strg+F9**). Die Farbe des Tracepoints ändert sich von Blau in Grau, wodurch gekennzeichnet wird, dass er deaktiviert wurde.

Alternativ dazu können Sie einen Breakpoint über das Dialgofeld "XSLT Breakpoints / Tracepoints" deaktivieren:

1. Wählen Sie die Menüoption **XSL/XQuery | Breakpoints/Tracepoints...**. Daraufhin wird das Dialogfeld XSLT Breakpoints / Tracepoints geöffnet, in dem Sie die aktuell definierten Breakpoints in allen offenen XML-Quelldokumenten und XSLT Stylesheets

sehen.



2. Entfernen Sie das Häkchen für die Tracepoints, die Sie deaktivieren möchten und klicken Sie zur Bestätigung auf **OK**. Diese Tracepoints werden nun statt blau grau angezeigt, d.h. sie sind deaktiviert.

So aktivieren Sie einen Tracepoint:

1. Platzieren Sie den Cursor in den Node oder Ausdruck, der den Tracepoint enthält.
2. Wählen Sie **XSL/XQuery | Tracepoint aktivieren/deaktivieren** (oder drücken Sie **Strg+F9**). Der Tracepoint wird nun statt grau wieder blau angezeigt, was bedeutet, dass er aktiviert wurde.

Suchen eines bestimmten Tracepoint

So suchen Sie einen bestimmten Tracepoint:

1. Wählen Sie die Menüoption **XSL/XQuery | Breakpoints/Tracepoints...**. Daraufhin wird das Dialogfeld "XSLT Breakpoints / Tracepoints" angezeigt.
2. Klicken Sie auf den gewünschten Tracepoint in der Liste.
3. Klicken Sie auf die Schaltfläche **Code bearbeiten**. Das Dialogfeld "XSLT Breakpoints / Tracepoints" wird geschlossen und der Textcursor erscheint direkt vor dem Tracepoint in der Textansicht des XSLT-Dokuments. In der Enhanced Grid-Ansicht wird die Tabellenzelle, die den Breakpoint enthält, rot markiert.

© 2007 Altova GmbH

12 XSLT/XQuery Profiler

Der XSLT/XQuery Profiler ist ein Tool zum Analysieren der Ausführungszeiten von XSLT (1.0 und 2.0) Stylesheets und XQuery-Dokumenten innerhalb von XMLSpy. Er gibt Auskunft darüber, wie lange die Ausführung der einzelnen Anweisungen in einem XSLT Stylesheet oder einem XQuery-Dokument jeweils dauert. Anhand dieser Informationen können Sie die Ausführungsdauer für diese Dateien optimieren.

Mit Hilfe des Profilers können Sie die Anweisungen ermitteln, deren Ausführung insgesamt am längsten dauert. Anhand dieser Informationen können Sie die Anweisungen optimieren. Die Ausführung mancher Anweisungen kann aus einem oder mehreren der folgenden Gründe lange dauern:

- die Anweisung ist zeitintensiv
- die Anweisung wird oft überprüft (hohe Trefferquote)

Profilerstellung für Trefferquote und Callgraph
Sie können im Profiler zwischen *Trefferquote* und *Callgraph* Profilerstellung auswählen. Bei beiden Profilerstellungsarten wird eine Statistik zur Ausführungszeit der einzelnen Anweisungen angezeigt.
Zu Optimierungszwecken wird normalerweise die Trefferquote-Profilerstellung verwendet. Dabei wird für jede Anweisung im Profiler eine Zeile angezeigt. Bei der Callgraph-Profilerstellung sehen Sie eine Aufzeichnung des Verlaufs der Ausführung der gesamten XSLT-Transformation oder XQuery-Ausführung. Hier sehen Sie, welche Templates/Funktionen während der Transformation aufgerufen werden und in welcher Reihenfolge dies geschieht. Im Ergebnis der Callgraph Profilerstellung wird jedesmal, wenn eine Anweisung aufgerufen wird, eine Zeile angezeigt, anstatt dass für jede Anweisung eine Zeile angezeigt wird.

Informationen zur Verwendung des XSLT/XQuery Profilers finden Sie unter Analysieren von XSLT-Transformationen oder Analysieren der XQuery-Ausführung.

Profiler-Ansichten
Die Ergebnisse der Analyse können in einer der folgenden Ansichten angezeigt werden. Klicken Sie dazu auf das entsprechende Register:

- **Listenansicht:** Die Statistik wird in Form einer Liste angezeigt, die z.B. nach Dauer der Anweisungsausführung oder der Dauer der Ausführung der Anweisungen und ihrer untergeordneten Nodes (Nachkommen) sortiert werden kann.

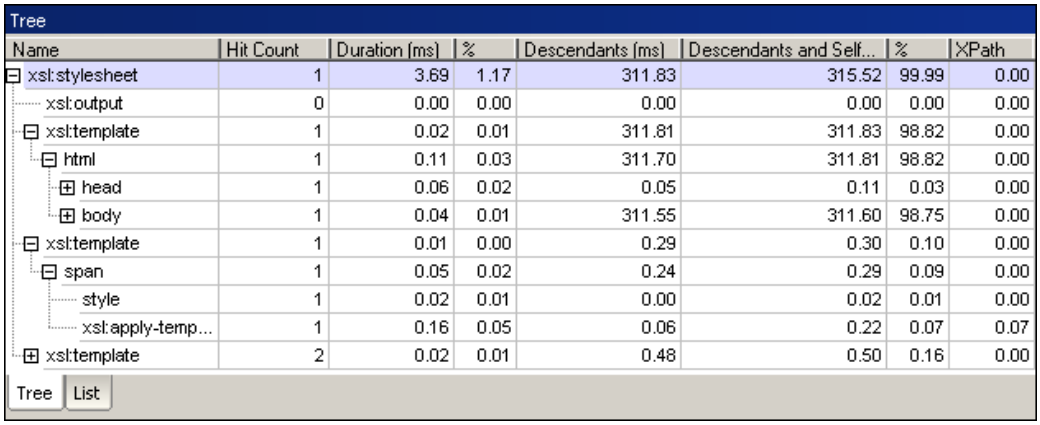
List

Index	Name	Hit Count	Duration [ms]	%	Des...	Descendants an...	%	XPath
0	xsl:stylesheet	1	3.69	1.17	311.83	315.52	99.99	0.00
1	xsl:output	0	0.00	0.00	0.00	0.00	0.00	0.00
2	xsl:template	1	0.02	0.01	311.81	311.83	98.82	0.00
3	html	1	0.11	0.03	311.70	311.81	98.82	0.00
4	head	1	0.06	0.02	0.05	0.11	0.03	0.00
5	title	1	0.05	0.02	0.00	0.05	0.02	0.00
6	body	1	0.04	0.01	311.55	311.60	98.75	0.00
7	xsl:for-each	1	0.22	0.07	311.33	311.55	98.74	0.18
8	xsl:for-each	1	0.18	0.06	1.30	1.49	0.47	0.17
9	div	1	0.05	0.02	1.25	1.30	0.41	0.00
10	style	1	0.03	0.01	0.00	0.03	0.01	0.00

Tree List

- **Baumstrukturansicht:** Die Statistik wird in Form einer Baumstruktur angezeigt. Sie sehen darin z.B. wie lange für die Ausführung einer Funktion gebraucht wurde. Sie

können die Struktur für diese Funktion anschließend erweitert anzeigen lassen und überprüfen, wie viel Zeit die Ausführung der einzelnen Anweisungen in Anspruch nahm und wie oft sie ausgeführt wurden.



Tree

Name	Hit Count	Duration (ms)	%	Descendants (ms)	Descendants and Self...	%	XPath
⊟ xsl:stylesheet	1	3.69	1.17	311.83	315.52	99.99	0.00
xsl:output	0	0.00	0.00	0.00	0.00	0.00	0.00
⊟ xsl:template	1	0.02	0.01	311.81	311.83	98.82	0.00
⊟ html	1	0.11	0.03	311.70	311.81	98.82	0.00
⊞ head	1	0.06	0.02	0.05	0.11	0.03	0.00
⊞ body	1	0.04	0.01	311.55	311.60	98.75	0.00
⊟ xsl:template	1	0.01	0.00	0.29	0.30	0.10	0.00
⊟ span	1	0.05	0.02	0.24	0.29	0.09	0.00
style	1	0.02	0.01	0.00	0.02	0.01	0.00
xsl:apply-temp...	1	0.16	0.05	0.06	0.22	0.07	0.07
⊞ xsl:template	2	0.02	0.01	0.48	0.50	0.16	0.00

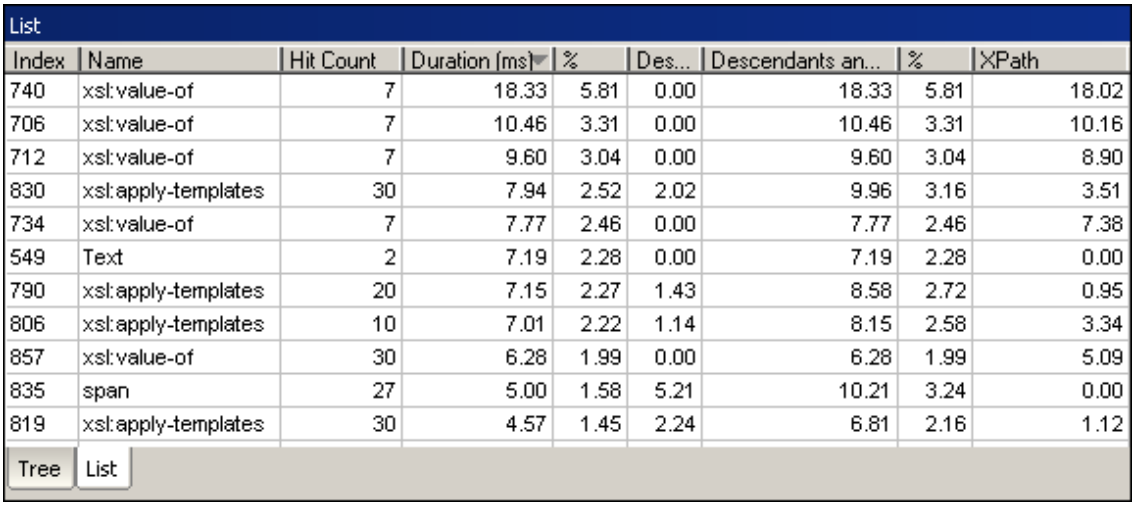
Tree | List

Sortieren von Ergebnissen
Nach Ausführung des Profilers können Sie die Ergebnisse nach der Ausführungsdauer der einzelnen Anweisungen oder nach der Anzahl der Aufrufe der Anweisung sortieren.

So sortieren Sie die Informationen im Profiler:

1. Klicken Sie auf das Register **Liste**.
2. Klicken Sie auf den Spaltenkopf der Spalte, nach der die Sortierung durchgeführt werden soll (z.B., **Anzahl der Treffer,** um nach der Anzahl der Aufrufe einer Anweisung zu sortieren oder **Dauer,** um nach der Ausführungsdauer der Anweisungen zu sortieren).

In diesem Screenshot sehen Sie den Inhalt des Profilers nach Anweisungsdauer in absteigender Reihenfolge sortiert.



List

Index	Name	Hit Count	Duration (ms)	%	Des...	Descendants an...	%	XPath
740	xsl:value-of	7	18.33	5.81	0.00	18.33	5.81	18.02
706	xsl:value-of	7	10.46	3.31	0.00	10.46	3.31	10.16
712	xsl:value-of	7	9.60	3.04	0.00	9.60	3.04	8.90
830	xsl:apply-templates	30	7.94	2.52	2.02	9.96	3.16	3.51
734	xsl:value-of	7	7.77	2.46	0.00	7.77	2.46	7.38
549	Text	2	7.19	2.28	0.00	7.19	2.28	0.00
790	xsl:apply-templates	20	7.15	2.27	1.43	8.58	2.72	0.95
806	xsl:apply-templates	10	7.01	2.22	1.14	8.15	2.58	3.34
857	xsl:value-of	30	6.28	1.99	0.00	6.28	1.99	5.09
835	span	27	5.00	1.58	5.21	10.21	3.24	0.00
819	xsl:apply-templates	30	4.57	1.45	2.24	6.81	2.16	1.12

Tree | List

Optimieren Ihrer XSLT Stylesheets und XQuery-Dokumente
Beachten Sie beim Optimieren der Ausführungsdauer von Anweisungen in XSLT Stylesheets und XQuery-Dokumenten die folgenden Richtlinien:

- Verwenden Sie in einer Anweisung nach Möglichkeit keine Variablen, wenn die Variable nur einmal verwendet wird, da die Initialisierung einer Variable zeitaufwändig sein kann.

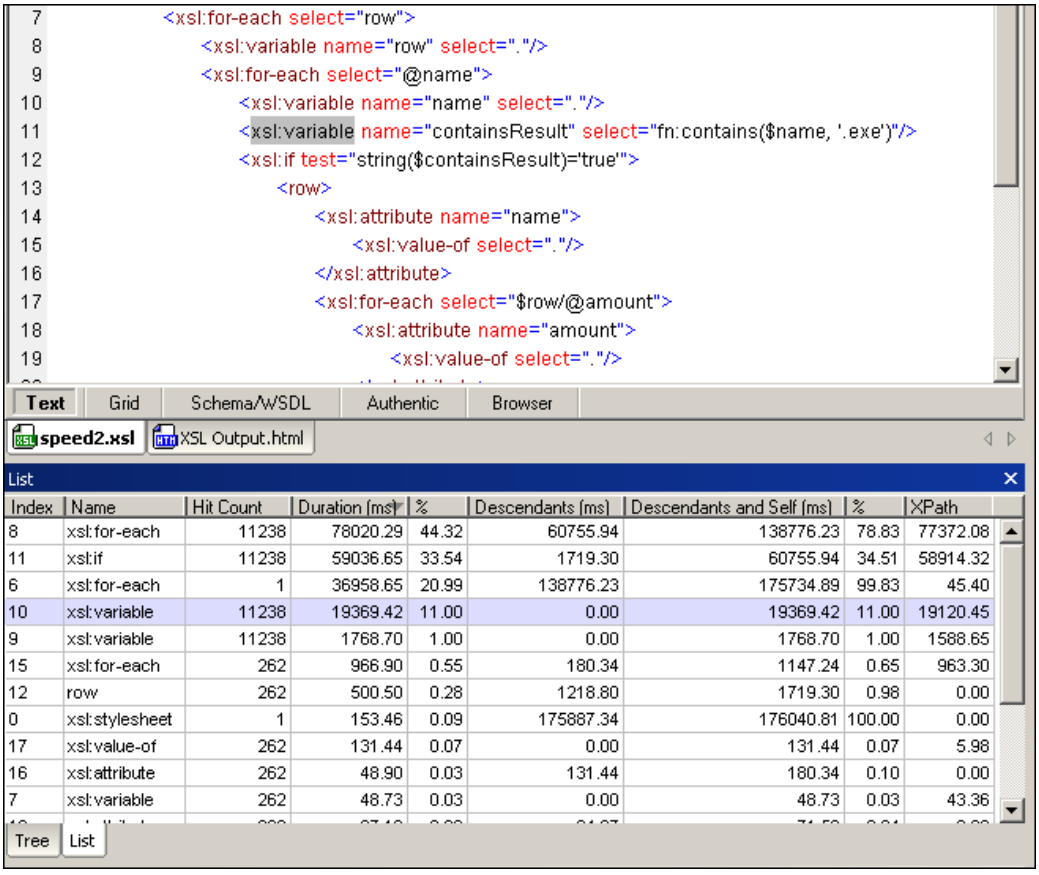
© 2007 Altova GmbH

In den folgenden XSLT Code-Fragmenten sehen Sie ein Beispiel, wie man Code durch Entfernen unnötiger Variablen optimieren kann. Beide Beispiele haben dasselbe Ergebnis. Im zweiten Beispiel werden jedoch die Variablen `name` und `containsResult` nicht verwendet:

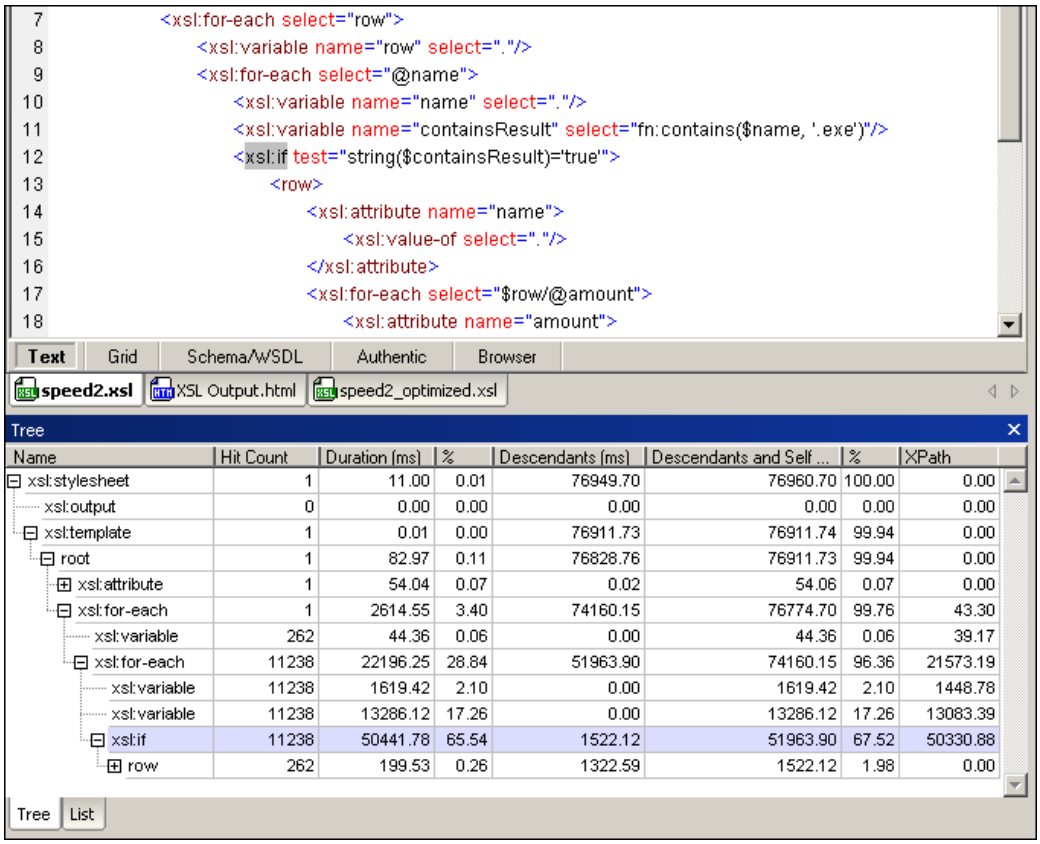
Code-Fragment 1:

```
<xsl:for-each select="row">
      <xsl:variable name="row" select="."/>
      <xsl:for-each select="@name">
            <xsl:variable name="name" select="."/>
            <xsl:variable name="containsResult" select="fn:
contains($name, '.exe')"/>
            <xsl:if test="string($containsResult)='true'">

            ...

            </xsl:if>
      </xsl:for-each>
</xsl:for-each>
```

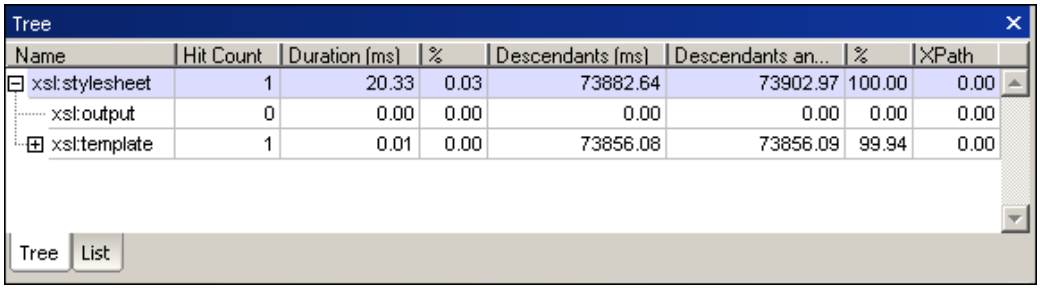
Im Screenshot unten sehen Sie das Ergebnis der Analyse der Datei, die dieses Codefragment enthält und zwar sortiert nach der Dauer der Ausführung der Anweisungen. Die Anweisung, in der die Variable containsResult initialisiert wird, benötigt etwa 19 Sekunden Gesamtausführungszeit.



Im Screenshot unten sehen Sie die Ergebnisse in der Baumstruktur. Hier sehen wir, dass die if-Anweisung, die die Variable containsResult verwendet, etwa 50 Sekunden Gesamtausführungszeit benötigt:



Die XSLT-Transformation dauert insgesamt etwa 74 Sekunden:

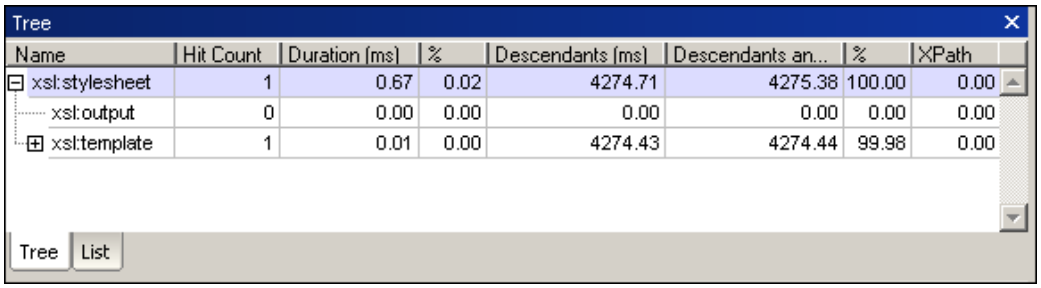


Code-Fragment 2:

```
<xsl:for-each select="row">
      <xsl:variable name="row" select="."/>
      <xsl:for-each select="@name">
            <xsl:if test="fn:contains(., '.exe')">

            ...

            </xsl:if>
      </xsl:for-each>
</xsl:for-each>
```

Nachdem das Stylesheet neu geschrieben wurde, ohne dass diese Variablen verwendet wurden, betrug die Gesamtausführungszeit nur mehr ca. 4,3 Sekunden:

© 2007 Altova GmbH



- Verwenden Sie Variablen, wenn ein Wert oder ein Ausdruck mehrmals verwendet wird.
- Vermeiden Sie es, lokale Konstanten-Variablen in einer Funktion zu erstellen; erstellen Sie stattdessen globale Konstanten-Variablen.
- Vermeiden Sie es, Konstanten-Struktur-Fragmente in einer Funktion zu erstellen; erstellen Sie diese stattdessen global.
- Verwenden Sie möglichst wenige Prädikate, da die Filterung mit Prädikaten für jeden Node separat ausgewertet wird. Sie können die Anzahl der Aufrufe von Prädikaten z. B. durch Vorfiltern mit Hilfe von Namen reduzieren. In diesem Beispiel wird bei zwei Prädikaten `*` verwendet:

```
//*[node-name()=Book][author="Steve"]
```

In dieser Anweisung, die dasselbe Resultat hat, werden der Name `Book` und nur ein Prädikat verwendet:

```
//Book[@Author="Steve"]
```

- Teilen Sie Anweisungen so auf, dass die Teile der Anweisung, die nur einmal ausgeführt werden müssen, nur einmal ausgeführt werden. Erstellen Sie globale Variablen anhand von Teilen, die nur vom globalen Kontext abhängen.

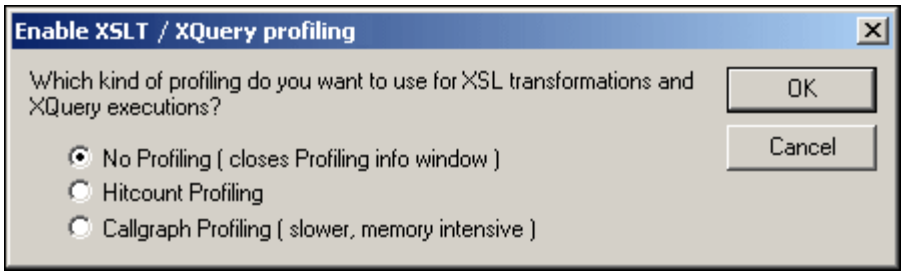
12.1 Analysieren von XSLT-Transformationen

Starten des Profilers

Bitte beachten Sie: Die Ausführungsdauer, die im Profiler angezeigt wird, kann auch von anderen Applikationen, die auf Ihrem Computer ausgeführt werden, beeinflusst werden. Beim Analysieren mit dem Profiler sollte am besten nur die XMLSpy Applikation geöffnet sein.

So analysieren Sie ein XSLT Stylesheet:

1. Öffnen Sie in XMLSpy die XML-Datei, die die Input-Daten für die XSLT-Transformation enthält.
2. Aktivieren Sie den Profiler durch Auswahl des Befehls **XSL/XQuery** | **XSLT aktivieren / XQuery Profilerstellung**. Es wird ein Dialogfeld geöffnet.

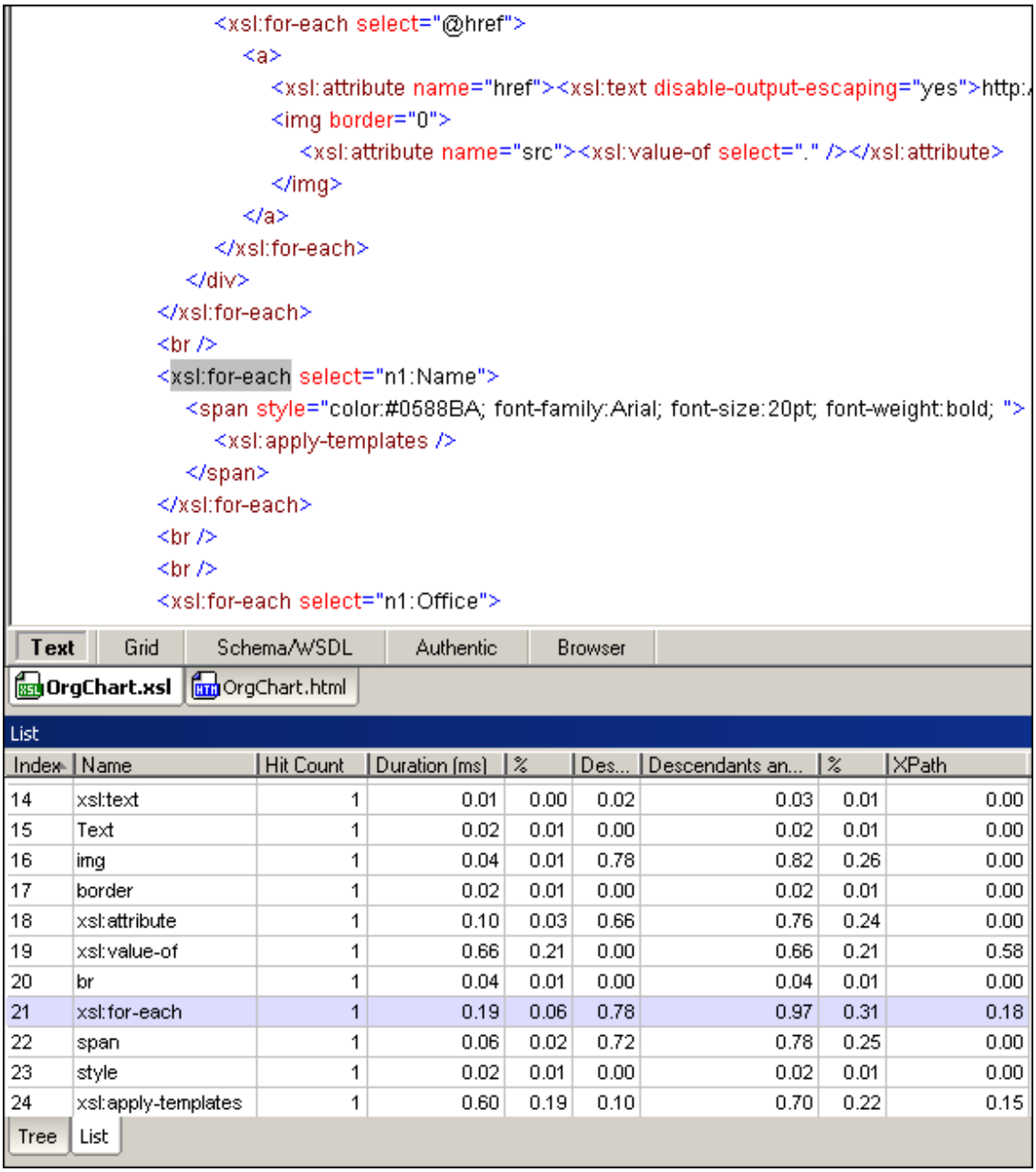


3. Wählen Sie die Option **Trefferquote Profilerstellung** oder **Callgraph Profilerstellung**. Klicken Sie zum Bestätigen auf **OK**. Es erscheint ein leeres Profiler-Fenster.
4. Führen Sie die XSL-Transformation aus (**XSL/XQuery | XSL-Transformation**). Es erscheint ein Dialogfeld, in dem Sie den Pfad zum XSLT Stylesheet angeben müssen, das Sie analysieren möchten. Nach Abschluss der Transformation wird im Profiler eine Statistik der Ausführungsdauer angezeigt.
5. Klicken Sie auf das "+"-Symbol, um die Zeilen im Profiler erweitert anzuzeigen, um die Ausführungsdauerstatistik für das XSLT Stylesheet zu sehen (*siehe Abbildung*). Beachten Sie, dass in diesen Abbildungen die Option "Trefferquote Profilerstellung" ausgewählt war.

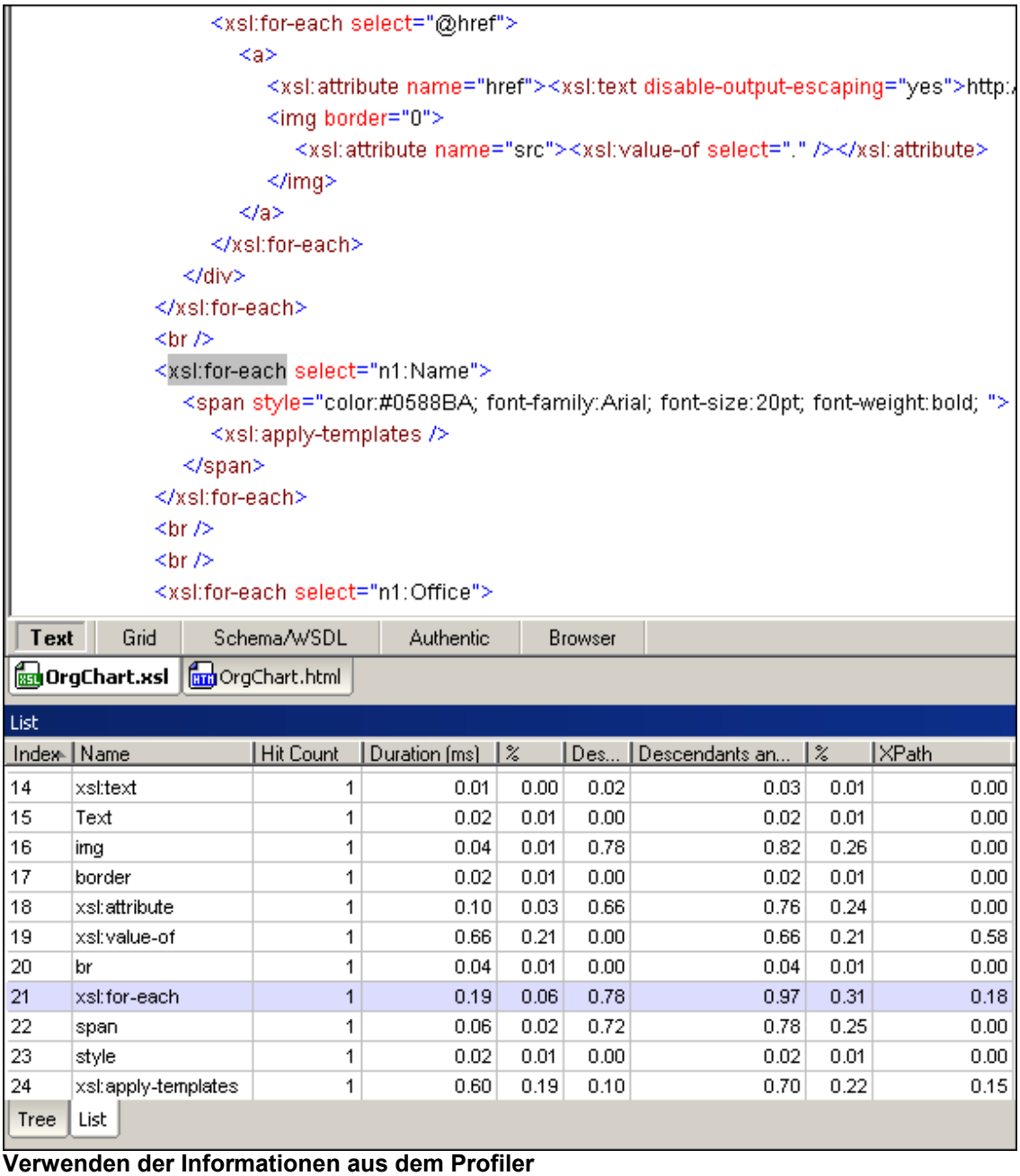
Klicken Sie auf eine Zeile im Profiler, um die entsprechende Anweisung in der analysierten Datei zu markieren.

In diesem Screenshot sehen Sie die Baumstrukturansicht im Profiler:

© 2007 Altova GmbH



Im folgenden Screenshot sehen Sie die Listenansicht im Profiler für dasselbe XSLT Stylesheet:



Verwenden der Informationen aus dem Profiler
Im Profiler werden die folgenden Informationen über die einzelnen Anweisungen im XSLT Stylesheet angezeigt:

- **Index:** Eine Nummer, die jeder Anweisung in der Reihenfolge, in der sie aufgerufen wurde, zugewiesen wird.
- **Name:** der Name der XSLT-Anweisung.
- **Anzahl der Treffer:** die Gesamtanzahl der Aufrufe der Anweisung während der Transformation.
- **Dauer (ms)** und **%:** Die Anzahl der Millisekunden, die für die Ausführung der Anweisung benötigt wurde, wobei dabei die Ausführungszeit für ihre untergeordneten Nodes nicht berücksichtigt wird, und der Prozentsatz an der Gesamtausführungsdauer.
- **Nachkommen (ms):** Die Anzahl der Millisekunden, die für die Ausführung der

© 2007 Altova GmbH

untergeordneten Nodes (Nachkommen) der Anweisung benötigt wurde.

- **Nachkommen und "selbst"** und **%:** Die Anzahl der Millisekunden, die für die Ausführung der Anweisung und der untergeordneten Nodes (Nachkommen) der Anweisung benötigt wurde und der Prozentsatz an der Gesamtausführungsdauer.
- **XPath:** Wenn die Anweisung eine XPath-Anweisung enthält, sehen Sie in dieser Spalte, wie lange die Ausführung der Anweisung dauerte.

Bitte beachten Sie: Bei Verwendung der Trefferquote-Profilerstellung entsprechen die Zeiten im Fenster "Profiler" der Gesamtausführungsdauer für alle Treffer in der Anweisung. Da bei Verwendung der Callgraph-Profilerstellung jeder Aufruf der Anweisung separat aufgelistet wird, beziehen sich die im Fenster "Profiler" angezeigten Zeiten auf die Dauer einer einzigen Ausführung der Anweisung.

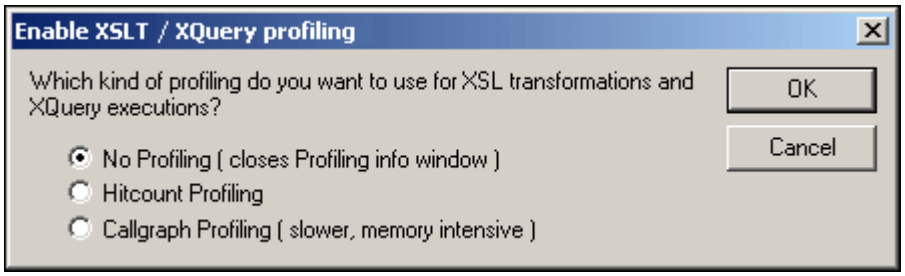
12.2 Analysieren einer XQuery-Ausführung

Starten des Profilers

Bitte beachten Sie: Die Ausführungsdauer, die im Profiler angezeigt wird, kann auch von anderen Applikationen, die auf Ihrem Computer ausgeführt werden, beeinflusst werden. Beim Analysieren mit dem Profiler sollte am besten nur die XMLSpy Applikation geöffnet sein.

So analysieren Sie ein XQuery-Dokument:

1. Öffnen Sie in XMLSpy die XML-Datei, die die Input-Daten für die XQuery-Ausführung enthält.
2. Aktivieren Sie den Profiler durch Auswahl des Befehls **XSL/XQuery** | **XSLT 2 aktivieren / XQuery Profilerstellung**. Es wird ein Dialogfeld geöffnet.

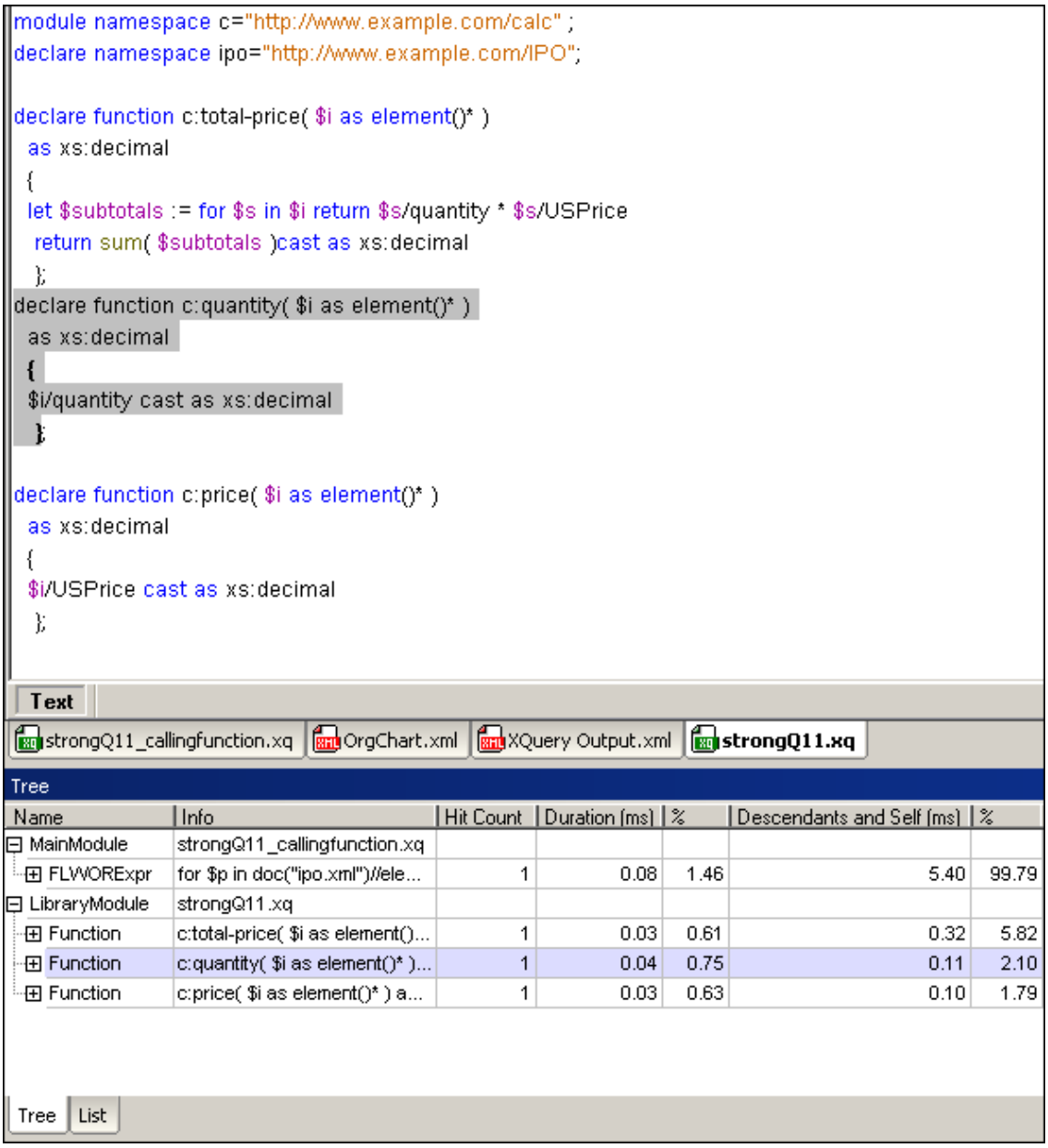


3. Wählen Sie die Option **Trefferquote Profilerstellung** oder **Callgraph-Profilerstellung**. Klicken Sie zum Bestätigen auf **OK**. Es erscheint ein leeres Profiler-Fenster.
4. Führen Sie die XQuery aus (**XSL/XQuery | XQuery-Ausführung**). Nach Abschluss der Ausführung wird im Profiler eine Statistik der Ausführungsdauer angezeigt.
5. Klicken Sie auf das "+"-Symbol, um die Zeilen im Profiler erweitert anzuzeigen, um die Ausführungsdauerstatistik für die Anweisungen im XQuery-Dokument zu sehen (*siehe Abbildung*). Beachten Sie, dass in diesen Abbildungen die Option **Trefferquote Profilerstellung** ausgewählt war.

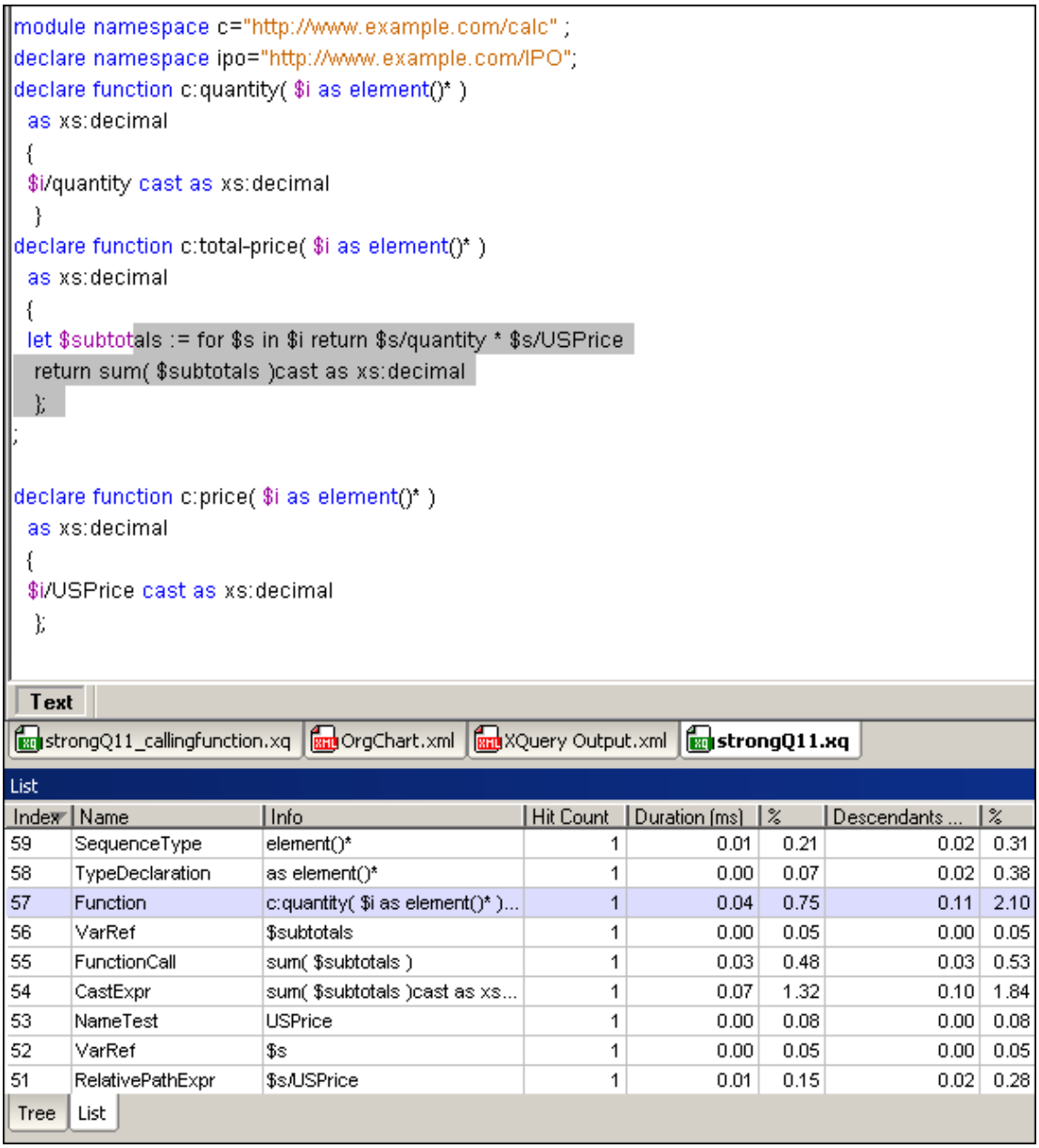
Klicken Sie auf eine Zeile im Profiler, um die entsprechende Anweisung in der analysierten Datei zu markieren.

In diesem Screenshot wird die Baumstrukturansicht im Profiler angezeigt.

© 2007 Altova GmbH



Im folgenden Screenshot sehen Sie die Listenansicht für dasselbe XQuery-Dokument:



Verwendung der Informationen im Profiler
Im Profiler werden die folgenden Informationen über die einzelnen Anweisungen im XQuery-Dokument angezeigt:

- **Index:** Eine Nummer, die jeder Anweisung in der Reihenfolge, in der sie aufgerufen wurde, zugewiesen wird.
- **Name:** der Name der XQuery-Anweisung.
- **Info:** Informationen über die Anweisung. Wenn es sich z.B. bei der Anweisung um eine Variablendeklaration handelt, enthält diese Spalte den Namen der Variable und ihren Wert; handelt es sich um eine Funktion, so enthält diese Spalte den Namen und die Parameter der Funktion.
- **Anzahl der Treffer:** die Gesamtanzahl der Aufrufe der Anweisung während der Ausführung.
- **Dauer (ms)** und **%:** Die Anzahl der Millisekunden, die für die Ausführung der

 © 2007 Altova GmbH

Anweisung ohne die untergeordneten Nodes benötigt wurde und der Prozentsatz an der Gesamtausführungsdauer.
- **Nachkommen und "selbst" (ms)** und **%:** Die Gesamtdauer der Ausführung der Anweisung und ihrer untergeodneten Nodes und der Prozentsatz an der Gesamtausführungsdauer.

Bitte beachten Sie: Bei Verwendung der Trefferquote-Profilerstellung entsprechen die Zeiten im Fenster "Profiler" der Gesamtausführungsdauer für alle Treffer in der Anweisung. Da bei Verwendung der Callgraph-Profilerstellung jeder Aufruf der Anweisung separat aufgelistet wird, beziehen sich die im Fenster "Profiler" angezeigten Zeiten auf die Dauer einer einzigen Ausführung der Anweisung.

13 SOAP Debugger

In diesem Abschnitt wird beschrieben wie man:

- SOAP Requests mit Hilfe des SOAP Debuggers sendet und empfängt
- Breakpoints zum Senden und Empfangen von SOAP Requests setzt
- einen fehlerhaften SOAP Request vor dem Absenden an einen Webservice bearbeitet

Beim Entwickeln von Applikationen, die Daten über einen Webservice aufrufen und diese weiterverarbeiten, kann es manchmal zu unerwarteten Problemen kommen. In diesem Kapitel wird erklärt, wie man die tatsächlich übertragenen Daten anzeigen und bearbeiten kann. Zu diesem Zweck wurde der SOAP Debugger entwickelt. Er fungiert als **Proxy Server** zwischen Client und Webservice. Die Funktionen des Debuggers gehen viel weiter in die Tiefe als die eines normalen Trace-Programms, das nur die Requests und Responses der einzelnen Calls speichert.

Mit Hilfe des SOAP Debuggers können Sie Breakpoints für jede Request und Response Message setzen und mittels XPath-Ausdrücken sogar bedingte Breakpoints setzen.

Der SOAP Debugger unterstützt:

- das schrittweise Überprüfen von SOAP Requests und Responses
- das Bearbeiten der SOAP Requests und Responses und
- das Weiterleiten bearbeiteter Requests an den Client oder Server.

Bitte beachten Sie:
Der SOAP Debugger unterstützt das sichere HTTP-Kommunikationsprotokoll **HTTPS** nicht.

© 2007 Altova GmbH

13.1 SOAP-Kommunikationsvorgang

Nach dem Starten des Proxy Servers (SOAP Debuggers) läuft der SOAP-Kommunikationsvorgang folgendermaßen ab:

- Der Proxy Server überwacht einen Eingang/Port ständig auf eingehende **Client Requests**

1. Die Client-Anwendung **sendet** einen Request an den **Proxy Server**
2. Client Requests können geändert werden, wenn/falls Breakpoints gefunden wurden.
3. Die **Request-Daten** des Proxy Servers werden an den **Webservice-Server** weitergeleitet.

- Der **Websevice Server antwortet** auf den Proxy Server Request und sendet die Response-Daten zurück an den Proxy Server.

4. Die Server Response-Daten können geändert werden, wenn/falls Breakpoints gefunden werden.
5. Die Proxy Server **Response-Daten** werden an die **Client-Applikation** weitergeleitet.

- Die Client Applikation empfängt Response-Daten vom Proxy Server.

Anwendungsbeispiel für Client- und Server-Applikationen:

- Die **Client-**Applikation, die SOAP-Messages sendet und empfängt, ist das Browser-Fenster
- Der "Nanonull Time Service" Dienst, fungiert als **Webservice Server**.
  Die URL für diesen Webservice lautet:
  http://www.nanonull.com/TimeService/TimeService.asmx?WSDL

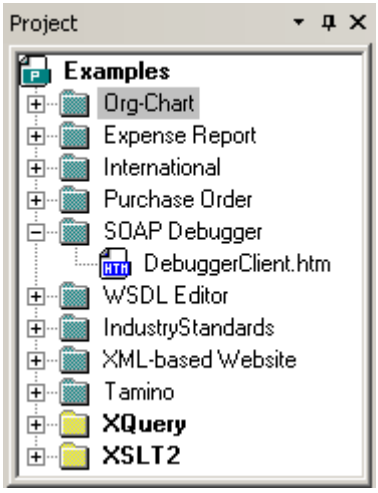
Port-Einstellungen
Der SOAP Debugger verwendet zum Überwachen der Client Request den Port 8080. Der Port kann nur beim Start einer neuen SOAP Debugger-Sitzung geändert werden.

Bitte beachten Sie, dass dieser Port durch Firewalls oder Werbefilter deaktiviert sein kann. Sie müssen diese daher zuerst deaktivieren oder einen anderen Port auswählen.

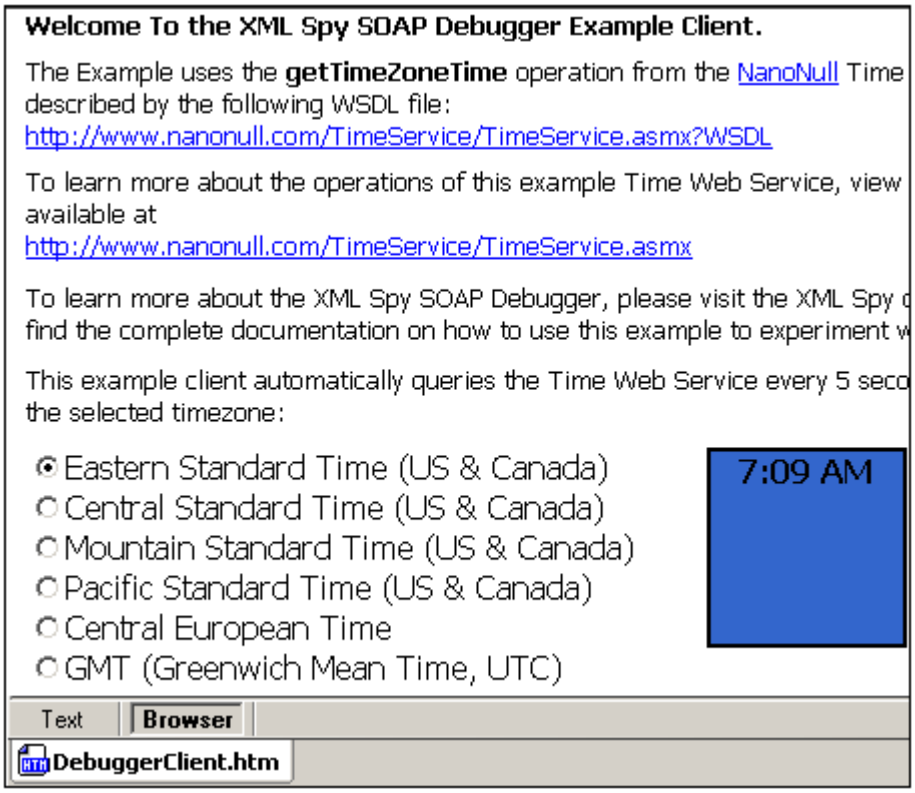
13.2 Verwendung des SOAP Debuggers

So verwenden Sie den SOAP Debugger:

1. Starten Sie XMLSpy.
2. Wählen Sie die Menüoption **Projekt | Öffnen**.
3. Wählen Sie die Datei "**Examples.spp**" im Ordner **XMLSpy2007\Examples** .
4. Klicken Sie auf das +-Zeichen des Ordners SOAP Debugger, um den Inhalt des Ordners anzuzeigen.



5. Doppelklicken Sie auf die Datei **DebuggerClient.htm,** um den SOAP Client zu starten und eine Verbindung zum Webservice herzustellen.



 © 2007 Altova GmbH

Im **html-**Fenster wird die aktuelle Eastern Standard Time (7:09 in diesem Beispiel) angezeigt.

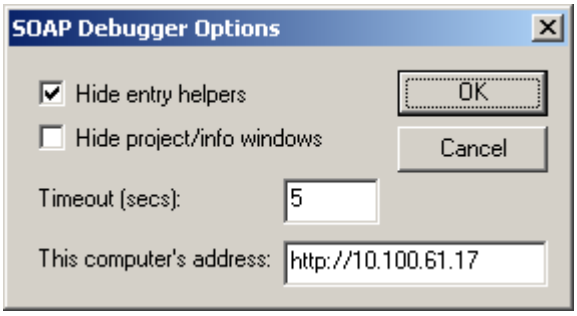
5. Klicken Sie auf das Optionsfeld GMT, um die aktuelle Greenwich Mean Time (westeuropäische Zeit) anzuzeigen.

An diesem Punkt ist offensichtlich ein Fehler aufgetreten. In der Anzeige der Uhrzeit sollte die korrekte Zeit angezeigt werden. Um die Ursache des Problems zu finden und beheben zu können, müssen wir den SOAP Debugger aktivieren, um die SOAP Messages zu analysieren.

Konfigurieren des SOAP Debuggers

So konfigurieren Sie den SOAP Debugger:

1. Wählen Sie die Menüoption **SOAP | SOAP Debugger-Optionen**.
2. Aktivieren Sie das Kontrollkästchen "Projekt-/Info-Fenster ausblenden" und klicken Sie auf OK.

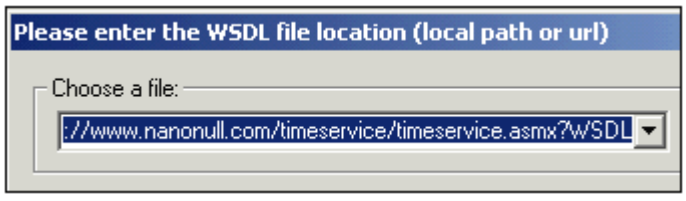


Auf diese Art haben Sie mehr Platz für das SOAP Request und das SOAP Response-Fenster. Stellen Sie sicher, dass Sie diese Einstellungen vor dem Starten der SOAP Debugger-Sitzung definieren.

Starten des SOAP Debuggers:

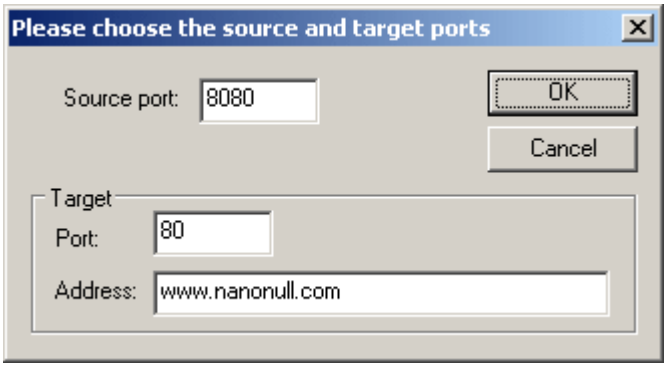
So starten Sie den SOAP Debugger:

1. Wählen Sie die Menüoption **SOAP | SOAP Debugger-Sitzung**.
   Daraufhin wird das Dialogfeld "Enter the WSDL file location..." angezeigt.

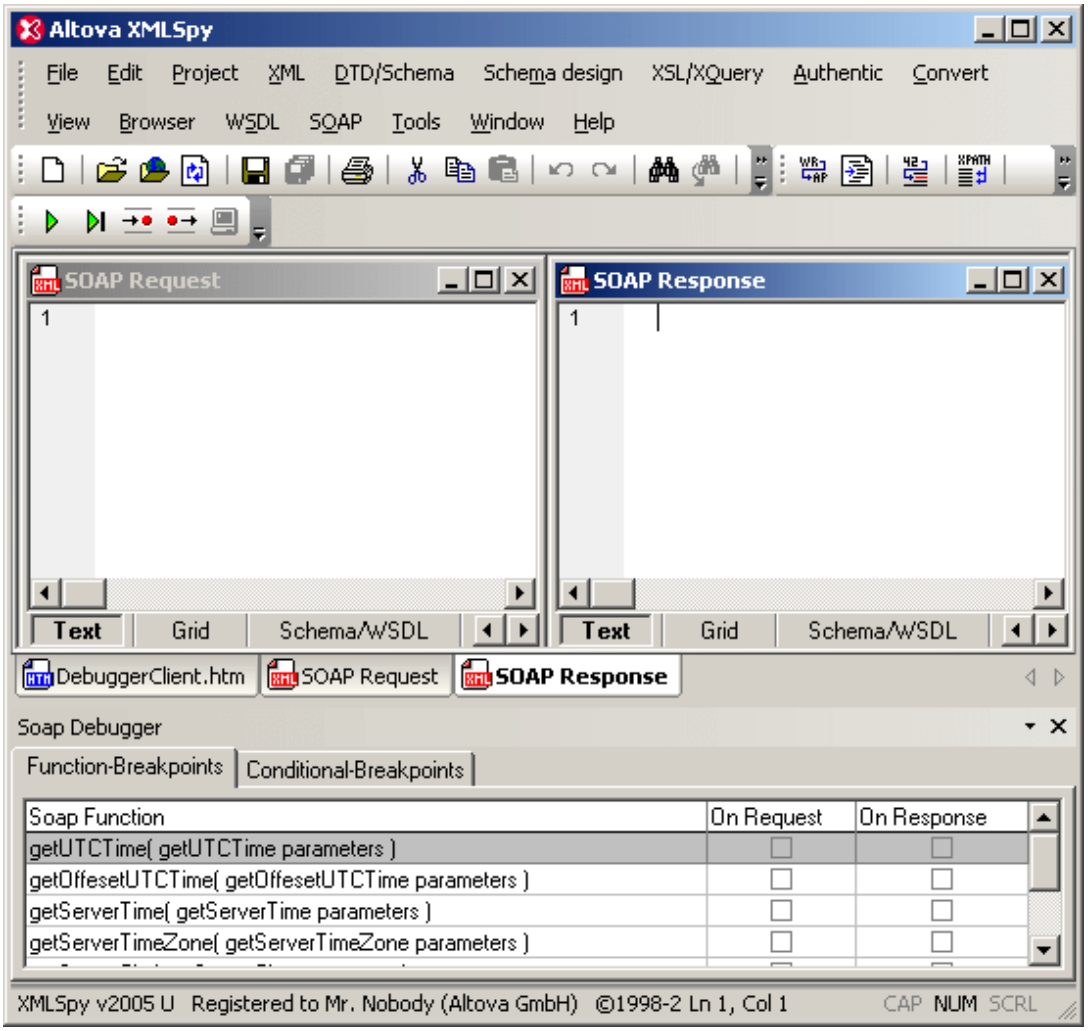


2. Geben Sie den Speicherort der WSDL-Datei ein (wie unten im Textfeld **http://www.nanonull.com/TimeService/TimeService.asmx?WSDL** gezeigt) und klicken Sie auf OK.
   Im folgenden Dialogfeld werden die Einstellungen für den Quell- und Zielport angezeigt. Die **Quell-Port-**Einstellung (8080) ist eine Standardeinstellung und wird automatisch eingegeben. Sie können diese Einstellung jedes Mal, wenn Sie den Debugger starten,

ändern.
Die Einstellung für den **Ziel-Port** stammt aus der WSDL-Datei und wird ebenfalls automatisch ausgefüllt.



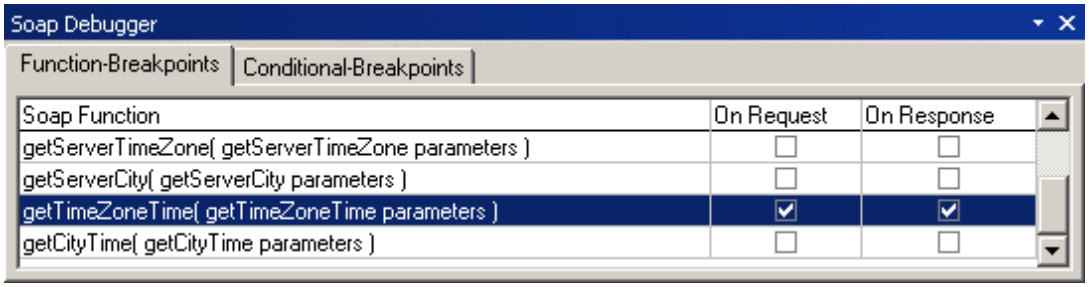
3. Klicken Sie auf die Schaltfläche **OK,** um die SOAP Debugger-Sitzung zu starten.



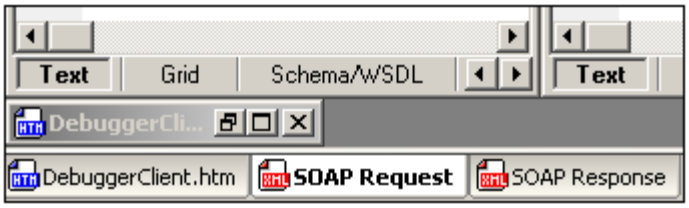
Daraufhin wird der SOAP Debugger Proxy Server im deaktivierten Status geöffnet. Sie sehen den Status daran, dass das Proxy Server-Symbol deaktiviert ist.

Setzen von Debugger Breakpoints
So setzen Sie Debugger Breakpoints:
Dieser Webservice verwendet die Methode **getTimeZoneTime**, um die Uhrzeit der Zeitzone zu suchen.

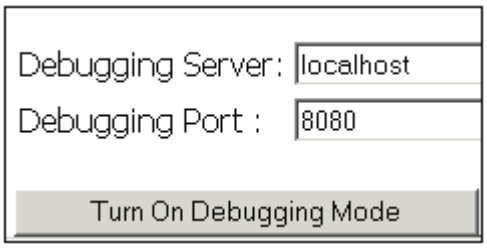
1. Aktivieren Sie die Kontrollkästchen **On Request** und **On Response** für diese Methode (**getTimeZoneTime)**. Auf diese Art können Sie die SOAP Requests und Responses auf Fehler analysieren.



2. Klicken Sie auf das **Start**-Symbol (oder wählen Sie den Menübefehl **SOAP | Starten**).
3. Klicken Sie auf das Register **DebuggerClient** um zum Client zu wechseln. Eventuell müssen Sie **Fenster | DebuggerClient.htm** auswählen, um den Inhalt zu sehen.

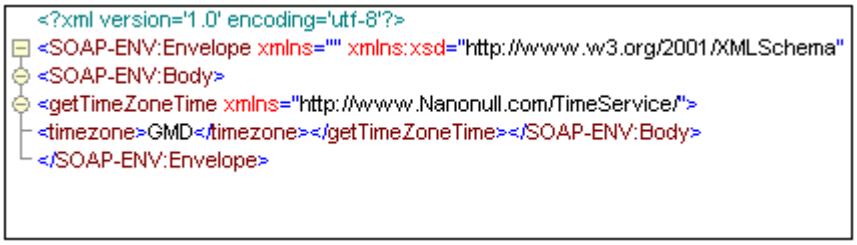


4. Klicken Sie im Client-Fenster auf die Schaltfläche "Turn On Debugging Mode".



Daraufhin wird die Meldung "Debug on" angezeigt und der SOAP Request wird an den Debugger gesendet.

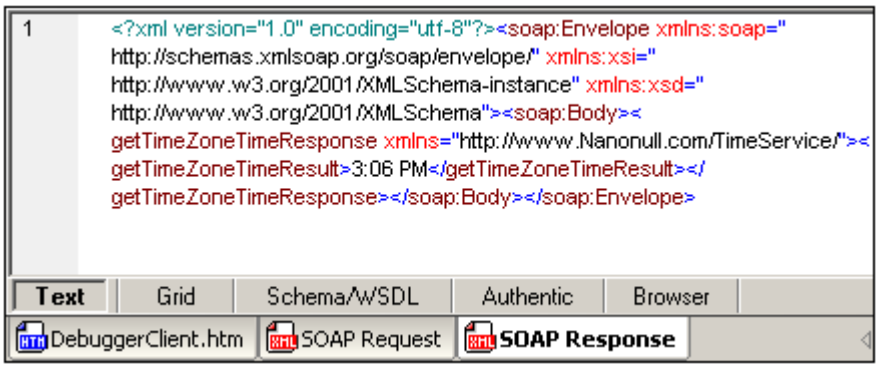
Anzeigen und Editieren des SOAP Request
Der SOAP Request wird automatisch im SOAP Request-Fenster des Debuggers in der Textansicht angezeigt. Wir wollen uns den Request nun ansehen und etwaige Fehler beheben.



Bei Element **timezone** stellen wir fest, dass der Wert **GMD** ist. Dies ist falsch und muss in GMT geändert werden.

1. Doppelklicken Sie in das Feld **timezone**, ändern Sie den Eintrag in **GMT** und drücken Sie die Eingabetaste.

2. Klicken Sie auf das **Start**-Symbol (oder wählen Sie den Menübefehl **SOAP | Starten**), um den **korrigierten Request** an den Webservice zu senden.
Nach wenigen Sekunden erscheint der **Webservice Response** auf den SOAP Request im SOAP Response-Fenster. Wählen Sie **Ansicht | Zeilenumbruch**, um den Inhalt zu sehen (siehe Diagramm unten).



Wir haben im Response keine Fehlermeldung erhalten, daher ist anzunehmen, dass es sich um die richtige Uhrzeit handelt.

3. Klicken Sie auf das Debugger **Start**-Symbol , um diesen Response an den **Client** zu senden (in diesem Fall das Browser-Fenster).
4. Klicken Sie auf das Register **DebuggerClient.htm**, um wieder zum Client zu wechseln und überprüfen Sie das neue Resultat.

Die Fehlermeldung wird nicht mehr angezeigt, stattdessen sehen Sie die korrekte

 © 2007 Altova GmbH

GMT-Zeit.

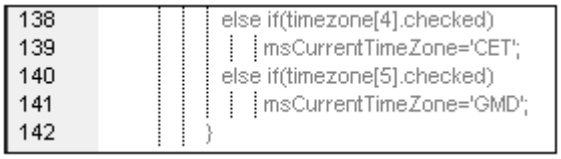
5. Wählen Sie den Menübefehl **SOAP | SOAP Debugger-Sitzung**, um die Debugger-Sitzung zu beenden.

Fehlerbehebung

Da wir nun wissen, dass der **Client** den unzulässigen Wert "GMD" anstatt "GMT" in seinem Request sendet und dass der Server auf "GMT" den korrekten Response senden würde, wissen wir, dass das Problem beim SOAP Client liegt und wir können das Problem nun beheben.

Gehen Sie im Fenster **DebuggerClient.html** folgendermaßen vor:

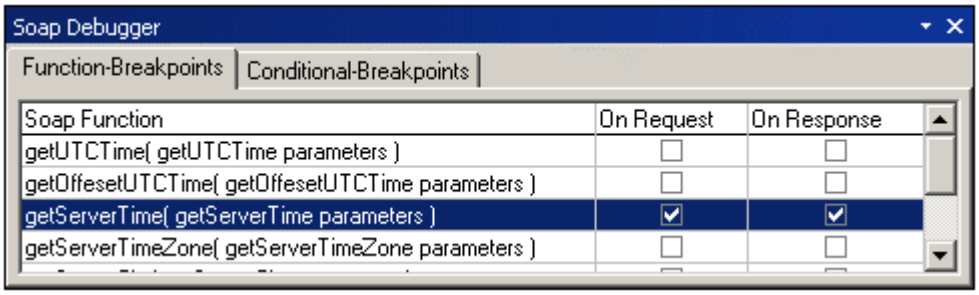
1. Klicken Sie auf die Schaltfläche "Text", öffnen Sie das Dialogfeld "Suchen" (**Bearbeiten | Suchen**) und suchen Sie nach "GMD".
   In XMLSpy wird das folgende Code-Fragment angezeigt:



2. Ändern Sie den Wert von "GMD" in "GMT" und wechseln Sie wieder in die Browser-Ansicht.
   Der SOAP Client arbeitet nun ordnungsgemäß, die Meldung "Unknown Time zone" wird nicht mehr angezeigt. Stattdessen erscheint im Fenster die richtige Uhrzeit. Somit ist das Problem behoben.

13.3 Setzen von Breakpoints im SOAP Debugger

Das Setzen und Löschen von Breakpoints erfolgt im SOAP Debugger-Fenster. Es besteht aus den folgenden zwei Registerkarten:



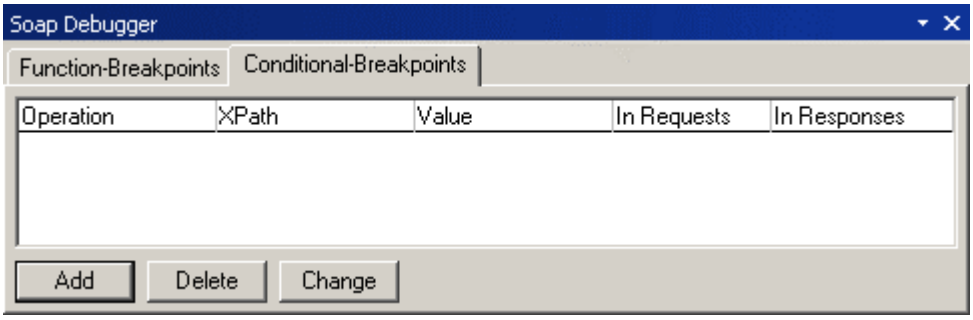
Function-Breakpoints
Das Register "Functional breakpoints" gestattet Ihnen, in Requests und/oder Responses auf SOAP-Methoden Breakpoints zu setzen. Der Debugger markiert die Funktionszeile, die den Breakpoint ausgelöst hat.

Datenpakete, die vom und an den Client gesendet werden, werden analysiert und den entsprechenden Funktionen aus der WSDL-Datei zugeordnet. Wenn ein Breakpoint für eine bestimmte Methode gesetzt wird, dann hält der SOAP Debugger an dieser Stelle an. Die Symbolleistenschaltflächen werden zu diesem Zeitpunkt aktiviert.

Die Daten werden in den Dokumentenfenstern "Soap Request" oder "Soap Response" angezeigt. Die in den SOAP-Fenstern angezeigten SOAP-Dokumente können nun geändert werden. Die Daten werden gesendet, sobald Sie auf eine der Symbolleistenschaltflächen (mit Ausnahme des Symbols "Proxyserver anhalten") klicken.

Conditional-Breakpoints
Auf dem Register "Conditional Breakpoints" können Sie XPath-Ausdrück verwenden, um Breakpoints zu definieren.

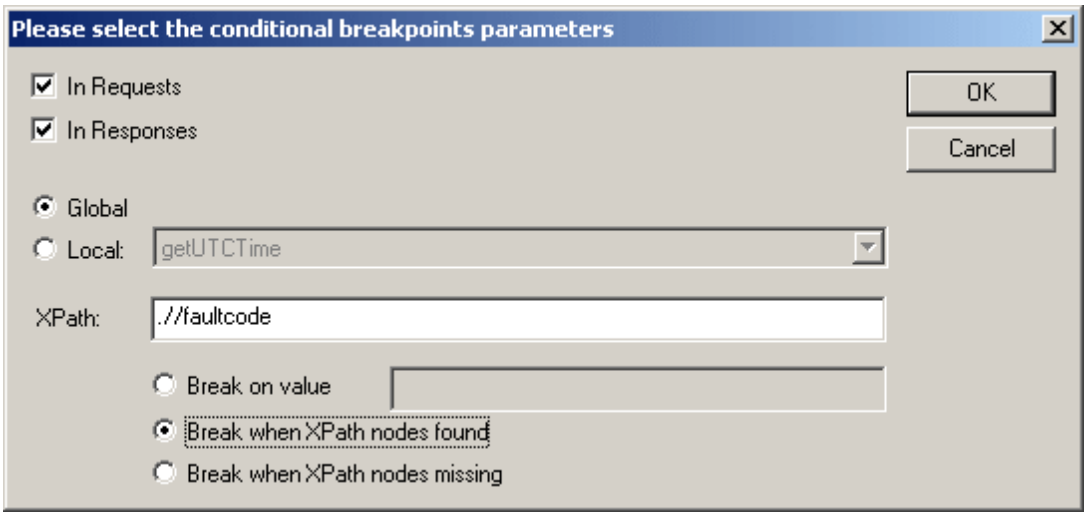
Wenn ein SOAP **Request** einen Fehler verursacht, muss der SOAP **Response** ein "faultcode"-Element enthalten. Wir möchten daher, dass an jedem faultcode-Element ein Breakpoint ausgelöst wird.



So fügen Sie einen bedingten Breakpoint (conditional Breakpoint) hinzu:

1. Klicken Sie auf das Register "Conditional Breakpoints" und anschließend auf die Schaltfläche **Add.**

 © 2007 Altova GmbH



2. Geben Sie den XPath-Ausdruck (z.B. **.//faultcode**) in das Feld XPath ein.
3. Klicken Sie auf das Optionsfeld "Anhalten wenn XPath Node gefunden".
4. Bestätigen Sie die Einstellungen durch Klicken auf OK.
   Der SOAP Debugger hält nun immer dann an, wenn im SOAP Request oder Response ein **.//faultcode-**Element erscheint.

Kontrollkästchen **Eintreffende Requests**
Mit Hilfe dieser Option können Sie die jeweilige Aktion definieren, die bei Auftreten eines Breakpoint in einem SOAP Request durchgeführt werden soll. Der Rest der Optionen des Dialogfelds bezieht sich auf den SOAP Request. Natürlich können Sie dieselben Einstellungen auch für SOAP Responses verwenden, wenn Sie auch das Kontrollkästchen "Eintreffende Responses" aktivieren.

Kontrollkästchen **Eintreffende Responses**
Mit Hilfe dieser Option können Sie die jeweilige Aktion definieren, die bei Auftreten eines Breakpoint in einem SOAP Response durchgeführt werden soll. Der Rest der Optionen des Dialogfeld bezieht sich auf den SOAP Response. Natürlich können Sie dieselben Einstellungen auch für SOAP Requests verwenden, wenn Sie auch das Kontrollkästchen "Eintreffende Requests" aktivieren.

Optionsfeld **Global**
Bei Auswahl der Option "Global" wird jede Methode/Funktion auf die von Ihnen definierte Bedingung durchsucht.

Optionsfeld **Lokal**
Bei Auswahl der Option "Lokal" müssen Sie eine bestimmte Methode/Funktion aus den Methoden/Funktionen der WSDL-Datei auswählen. Diese Methode/Funktion wird dann nach der von Ihnen definierten Bedingung durchsucht.

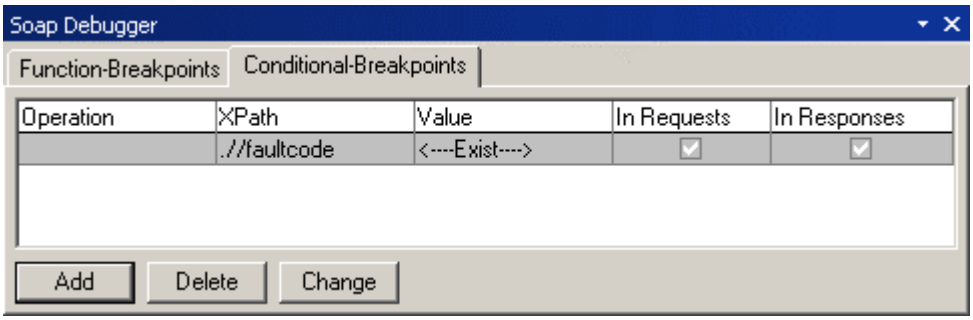
Feld **XPath**
Geben Sie hier den gewünschten XPath-Ausdruck/Node ein. Sie müssen hier einen XPath eingeben, um eine der Optionen der Optionsfelder verwenden zu können.

Optionsfeld **Anhalten bei Wert**
Der Debugger hält an, wenn einer der XPath Nodes mit dem in diesem Feld angegebenen Wert übereinstimmt.

Optionsfeld **Anhalten wenn XPath Node gefunden**
Der Debugger hält an, wenn der XPath Node **.//faultcode**, der im XPath-Feld eingegeben

wurde, im SOAP Request oder dem SOAP Response **vorhanden** ist.

Optionsfeld **Anhalten bei fehlenden XPath Nodes**
Der Debugger hält an, wenn der XPath Node **.//faultcode**, der im XPath-Feld eingegeben wurde, im SOAP Request oder dem SOAP Response **nicht vorhanden** ist.



Operation
Die Spalte "Operation" enthält die Methode, nach der gesucht wird. Wenn Sie das Optionsfeld **Global** aktiviert haben, bleibt dieses Feld leer. Bei Auswahl von **Lokal** wird die gewählte Methode hier angezeigt.

XPath
Diese Spalte enthält den von Ihnen definierten XPath-Ausdruck.

Value
Diese Spalte enthält den XPath-Wert, auf den die XPath Nodes überprüft werden. Wenn Sie die Option "Anhalten bei Wert" gewählt haben, wird der von Ihnen eingegebene String hier angezeigt. Wenn Sie die Option Anhalten wenn XPath Node gefunden" gewählt haben, so wird <--Exist--> angezeigt. Wenn Sie die Option "Anhalten bei fehlenden XPath Nodes" gewählt haben, wird <--Missing--> angezeigt.

In Requests
Das Kontrollkästchen wird angezeigt, wenn der XPath-Ausdruck im SOAP Request mit einem Häkchen versehen ist.
Sie können die Einstellungen ändern, indem Sie das Kontrollkästchen direkt in der Spalte aktivieren.

In Responses
Das Kontrollkästchen wird angezeigt, wenn der XPath-Ausdruck im SOAP Response mit einem Häkchen versehen ist.
Sie können die Einstellungen ändern, indem Sie das Kontrollkästchen direkt in der Spalte aktivieren.

So bearbeiten Sie die bedingten Breakpoints:

- Doppelklicken Sie auf die entsprechende Zeile im Register oder
- Klicken Sie auf die Schaltfläche "Change".

So löschen Sie einen bedingten Breakpoint

- Klicken Sie auf die zu löschende Zeile und anschließend auf die Schaltfläche "Delete".

© 2007 Altova GmbH

14 Code Generator

XMLSpy includes a built-in code generator which can automatically generate Java, C++ or C# class files from XML Schema definitions

XML is not a full programming language in that it cannot be compiled or executed as a stand-alone binary executable file; rather XML documents must be bound to an external software application or runtime environment such as a business-to-business application or Web service.

The implementation of any custom XML software application ultimately requires writing programmatic access methods within your code to create, validate, process, transform, modify or perform any in-memory operation on an XML document.

Program coding is most commonly done through the use of high-level XML processing Application Program Interfaces (API) such as Microsoft MSXML and Apache Xerces Version 2.6.0 and higher, which are freely available for various programming languages.

14.1 Introduction to code generator

In the case of XML Schemas the XMLSpy code generator's default templates automatically generate class definitions corresponding to all declared elements or complex types which redefine any complex type in your XML Schema, preserving the class derivation as defined by extensions of complex types in your XML Schema.

In the case of complex schemas which import schema components from multiple namespaces, XMLSpy preserves this information by generating the appropriate C#, or C++ namespaces or Java packages.

Additional code is implemented, such as functions which read XML files into a Document Object Model (DOM) in-memory representation, write XML files from a DOM representation back to a system file, or to convert strings to XML DOM trees and vice versa.

The output program code is expressed in C++, Java or C# programming languages.

Target Language	C++	C#	Java
Development environments	Microsoft Visual C++ 2005 Microsoft Visual C++ .NET 2003 Microsoft Visual C++ 6.0	Microsoft Visual Studio 2005 Microsoft Visual Studio .NET 2003 Microsoft Visual Studio .NET 2002 Borland C#Builder Mono (Makefile)	Borland JBuilder Apache Ant (build.xml file) Eclipse 3.x
XML DOM implementations	MSXML 4.0 or Apache Xerces 2.6 or later	System.Xml	JAXP
Database API (MapForce only)	ADO	ADO.NET	JDBC

C++
The C++ generated output uses either MSXML 4.0, or Apache Xerces 2.6 or later. Both MapForce and XMLSpy generate complete project and solution/workspace files for Visual C++ 6.0, Visual C++ .NET 2003 and 2005 directly. **.sln** and .**vcproj** files are generated in addition to the **.dsw** /**.dsp** files for Visual Studio 6.0. The generated code optionally supports MFC, e.g. by generated CString conversion methods.

Please note:
When building C++ code for Visual Studio 2005 and using a Xerces library precompiled for Visual C++ 6.0, a compiler setting has to be changed in all projects of the solution:

1. Select all projects in the Solution Explorer.
2. [Project] | [Properties] | [Configuration Properties] | [C/C++] | [Language]
3. Select *All Configurations*
4. Change **Treat wchar_t** as **Built-in Type** to **No (/Zc:wchar_t-)**

C#
The generated C# code uses the .NET XML classes (System.Xml) and can be used from any .NET capable programming language, e.g. VB.NET, Managed C++, J# or any of the several languages that target the .NET platform. Project files can be generated for Visual Studio .NET, 2003, 2005 and Borland C#Builder.

Java
The generated Java output is written against the industry-standard Java API for XML Parsing (JAXP) and includes a JBuilder project file, an ANT build file and project files for Eclipse. Java

 © 2007 Altova GmbH

1.4 or higher is supported, Java 5 or higher is recommended.

Generated output in XMLSpy:

Generated output	Location	XMLSpy
Standard libraries	Altova folder	☑
Schema wrapper libraries	Schema name folder	☑
Application		
Test Application skeleton - source code with samples	The test application must be edited for it to function. Not functional at this point.	☑

Code generator templates
Output code is completely customizable via a simple yet powerful template language which gives full control in mapping XML Schema built-in data-types to the primitive datatypes of a particular programming language.
It allows you to easily replace the underlying parsing and validating engine, customize code according to your company's writing conventions, or use different base libraries such as the Microsoft Foundation Classes (MFC) and the Standard Template Library (STL).

Additionally you can build your own templates to automate the generation of anything imaginable, for example: EJB's, WSDL files, SQL scripts, ASP or WML code.

XMLSpy's schema editor is well suited as a software modeling and prototyping tool, allowing XML applications to be rapidly prototyped at a high level in XML Schema and then automatically generated. Changes to an application's XML Schema content model can be immediately reconciled with a software implementation simply by re-running the code generator.

14.2 What's new in version 2007r3

Code Generator has been redesigned for version 2007 release 3 to simplify usage of the generated code, reduce code volume and increase performance.

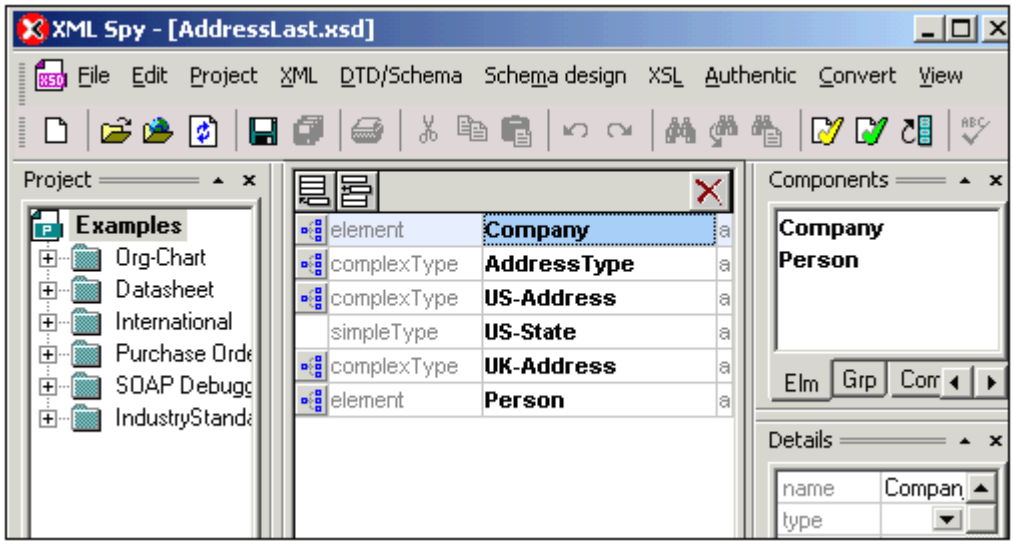
- Handling of XML documents and nodes with explicit ownership, to avoid memory leaks and to enable multi-threading
- New syntax to avoid name collisions
- New data types for simpler usage and higher performance (native types where possible, new null handling, ...)
- Attributes are no longer generated as collections
- Simple element content is now also treated like a special attribute, for consistency
- New internal object model (important for customized SPL templates)
- Compatibility mode to generate code in the style of older releases
-

© 2007 Altova GmbH

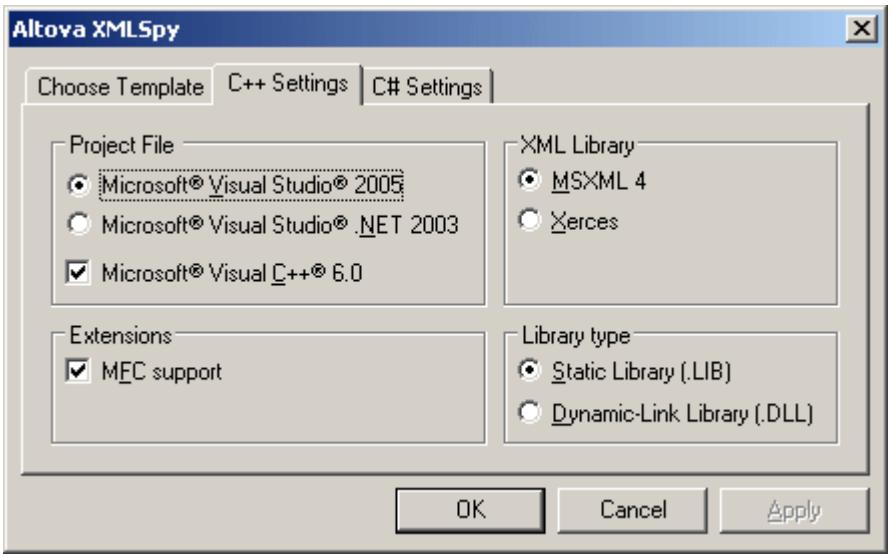
14.3 Generating source code from a schema

The following sequence shows how to generate the source code produced by the AddressLast schema file as the input.

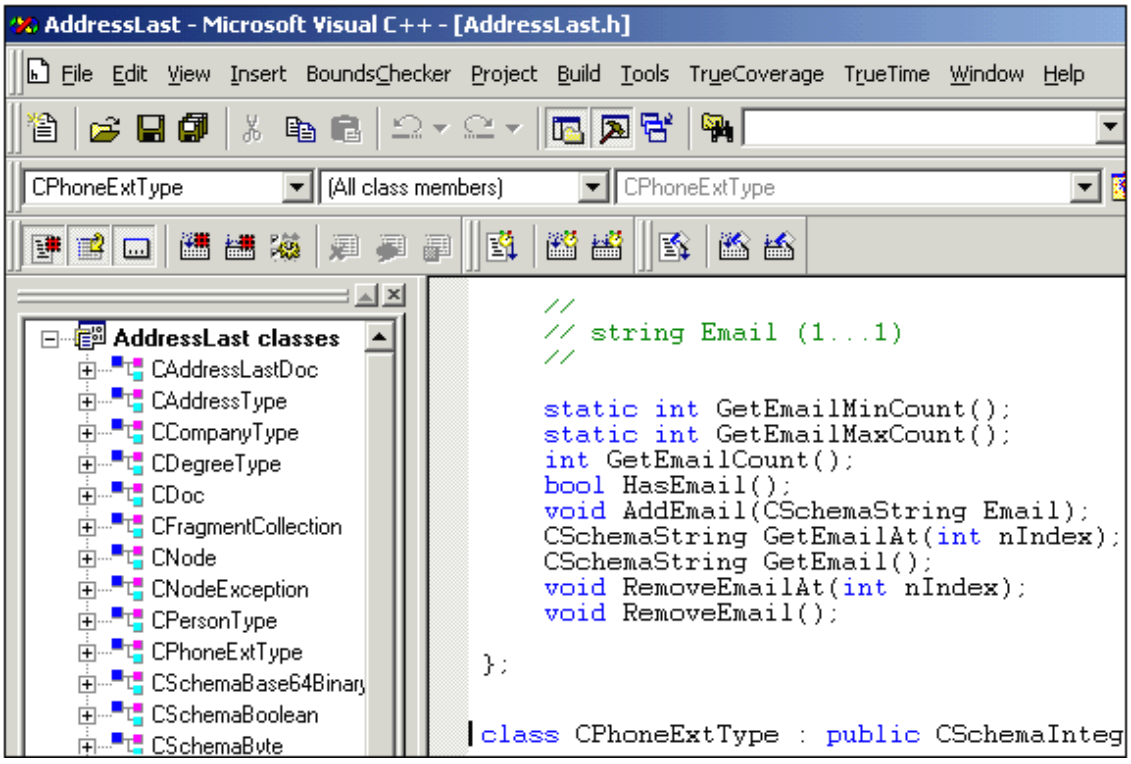
1. Open the schema you want to generate source code for (e.g., AddressLast).



2. Select the menu item **DTD/Schema | Generate Program Code...**.
3. Select the type of code you want to generate, Java, C++ or C#, (e.g., java.spl, cpp.spl or cs.spl). The C++ Settings tab allows you to further specify the code generation settings for C++.



4. Click **OK**. The Browse for folder dialog appears.
5. Select the target folder and click **OK**.
6. You are prompted to open the newly created project in Microsoft Visual Studio. Click **Yes**.
   If Java code is produced, you are prompted to open the corresponding output directory.



7. The generated workspace contains 4 projects: Altova, AltovaXML, AddressLast, and AddressLastTest. The latter is a skeleton application to test your new generated library. Open the AddressLastTest.cpp file and write your test code into the Example() function.

 © 2007 Altova GmbH

14.4 Using the generated code library

Overview
The library generated by XMLSpy is a DOM (W3C Document Object Model) wrapper that allows you to read, modify and create XML documents easily and safely. All data is held inside the DOM, and there are methods for extracting data from the DOM, and to update and create data into the DOM.

A different approach to code generation for XML would be to create value classes with serialization methods, where the data is stored inside the generated classes and XML is only used for saving and loading. This is not supported out-of-the-box because it is less flexible, but it is possible using custom SPL templates.

Compatibility Mode
Depending on the value of the setting "Generate code compatible to", the generated classes have different names, members and methods. The following description applies to **version 2007r3 only**. For the other settings please refer to the chapter about using generated code compatible to old versions.

Generated Libraries
When XMLSpy generates code from an XML Schema or DTD, the following libraries are generated:

Name	Purpose
Altova	Base library containing common runtime support, identical for every schema
AltovaXML	Base library containing runtime support for XML, identical for every schema
YourSchema	Library containing declarations generated from the input schema, named as the schema file or DTD
YourSchema Test	Test application skeleton (XMLSpy only)

The test application skeleton is a compilable application that calls an empty Example() method. You can add your test code into this method for easy and quick testing of your new generated library.

Name generation and namespaces
Generally, the code generator tries to preserve the names for generated namespaces, classes and members from the original XML Schema. Characters that are not valid in identifiers in the target language are replaced by a "_". Names that would collide with other names or reserved words are made unique by appending a number. Name generation can be influenced by changing default settings in the SPL template.

The namespaces from the XML Schema are converted to packages in Java or namespaces in C# or C++ code, using the namespace prefix from the schema as code namespace. The complete library is enclosed in a package or namespace derived from the schema file name, so you can use multiple generated libraries in one program without name conflicts.

Data Types
XML Schema has a much more elaborate data type model than Java, C# or C++. Code Generator converts the 44 XML Schema types to a couple of language-specific primitive types, or to classes delivered with the Altova library. The mapping of simple types can be configured in the SPL template.

Complex types and derived types defined in the schema are converted to classes in the generated library.

Generated Classes
XMLSpy generally generates one class per type found in the XML Schema, or per element in the DTD. One additional class per library is generated to represent the document itself, which can be used for loading and saving the document.

Memory Management
A DOM tree is comprised of nodes, which are always owned by a specific DOM document - even if the node is not currently part of the document's content. All generated classes are references to the DOM nodes they represent, not values. This means that assigning a generated class does not copy the value, it only creates an additional reference to the same data.

14.4.1 Example schema

Using the generated classes
The previous sections explained how to open a schema, choose a template and generate the files you need. You then open the generated project or the corresponding directory and get down to work.

To keep things simple, we will work with a very small xsd file, the source code is shown below. Save this as **Library.xsd** if you want to get the same results as our example:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="http://www.nanonull.com/LibrarySample" xmlns:xs="
http://www.w3.org/2001/XMLSchema" xmlns="http://www.nanonull.com/LibrarySample"
elementFormDefault="qualified" attributeFormDefault="unqualified">
    <xs:element name="Library">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="Book" type="BookType" minOccurs="0" maxOccurs="
unbounded"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:complexType name="BookType">
        <xs:sequence>
            <xs:element name="Title" type="xs:string"/>
            <xs:element name="Author" type="xs:string" maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="ISBN" type="xs:string" use="required"/>
    </xs:complexType>
</xs:schema>
```

We will only use this schema file in this example, it will therefore be easy to figure out the differences between the different programming languages in the following code examples.

Please note that the generated classes will be named differently from xsd to xsd file. When working with your own schemas or other samples, make sure you adapt the names of the

 © 2007 Altova GmbH

functions and variables.

14.4.2 Using the generated Java library

Please note: This topic describes code generation when using the option **Generate code compatible to V2007r3** in the code generation dialog.

Generated Packages

Name	Purpose
com.altova	Base package containing common runtime support, identical for every schema
com.altova.xml	Base package containing runtime support for XML, identical for every schema
com.YourSchema	Package containing declarations generated from the input schema, named as the schema file or DTD
*com.YourSchema*Test	Test application skeleton (XMLSpy only)

DOM Implementation

The generated Java code uses JAXP (Java API for XML Processing) as the underlying DOM interface. Please note that the default DOM implementation delivered with Java 1.4 does not automatically create namespace attributes when serializing XML documents. If you encounter problems, please update to Java 5.

Data Types

The default mapping of XML Schema types to Java data types is:

XML Schema	Java	Remarks
xs:string	String	
xs:boolean	boolean	
xs:decimal	java.math.BigDecimal	
xs:float, xs:double	double	
xs:integer	java.math.BigInteger	
xs:long	long	
xs:unsignedLong	java.math.BigInteger	Java does not have unsigned types.
xs:int	int	
xs:unsignedInt	long	Java does not have unsigned types.
xs:dateTime, date, time, gYearMonth, gYear, gMonthDay, gDay, gMonth	com.altova.types.DateTime	
xs:duration	com.altova.types.Duration	
xs:hexBinary and xs:base64Binary	byte[]	Encoding and decoding of binary data is done automatically.
xs:anySimpleType	string	

All XML Schema types not contained in this list are derived types, and mapped to the same Java type as their respective base type.

Generated Document Class
In addition to the classes for the types declared in the XML Schema, the document class is generated. It contains all possible root elements as members, and in addition the following methods ("Doc" stands for the name of the generated document class itself):

Method	Purpose
static *Doc* **loadFromFile**(String fileName)	Loads an XML document from a file
static *Doc* **loadFromString**(String xml)	Loads an XML document from a string
static *Doc* **loadFromBinary**(byte[] xml)	Loads an XML document from a byte array
void **saveToFile**(String fileName, boolean prettyPrint, String encoding = "UTF-8")	Saves an XML document to a file with the specified encoding. If no encoding is specified, UTF-8 is assumed.
String **saveToString**(boolean prettyPrint)	Saves an XML document to a string
byte[] **saveToBinary**(boolean prettyPrint, String encoding = "UTF-8")	Saves an XML document to a byte array with the specified encoding. If no encoding is specified, UTF-8 is assumed.
static *Doc* **createDocument**()	Creates a new, empty XML document.
void **setSchemaLocation**(String schemaLocation)	Adds an xsi:schemaLocation or xsi:noNamespaceSchemaLocation attribute to the root element. A root element must already exist.

Generated Member Attribute Class
For each member attribute of a type, a class with the following methods is created.

Method	Purpose
boolean **exists**()	Returns true if the attribute exists
void **remove**()	Removes the attribute from its parent element
AttributeType **getValue**()	Gets the attribute value
void **setValue**(*AttributeType* value)	Sets the attribute value

Generated Member Element Class
For each member element of a type, a class with the following methods created.

Method	Purpose
MemberType **first**()	Returns the first instance of the member element
MemberType **at**(int index)	Returns the member element specified by the index
MemberType **last**()	Returns the last instance of the member element
MemberType **append**()	Creates a new element and appends it to its parent
boolean **exists**()	Returns true if at least one element exists
int **count**()	Returns the count of elements
void **remove**()	Deletes all occurrences of the element from its parent
java.util.Iterator **iterator**()	Returns an object for iterating instances of the member element.
MemberType **getValue**()	Gets the element content (only generated if element can have simple or mixed content)

 © 2007 Altova GmbH

void **setValue**(*MemberType* value)	Sets the element content (only generated if element can have simple or mixed content)

Creating XML Documents

The generated code library makes it very easy to create XML files. Let's start with an example XML file for the example schema:

```
<?xml version="1.0" encoding="UTF-8"?>
<Library xsi:schemaLocation="http://www.nanonull.com/LibrarySample Library.xsd" xmlns="
http://www.nanonull.com/LibrarySample" xmlns:xsi="
http://www.w3.org/2001/XMLSchema-instance">
    <Book ISBN="0764549642">
        <Title>The XML Spy Handbook</Title>
        <Author>Altova</Author>
    </Book>
</Library>
```

The following Java code was used to create this file. The classes Library2, LibraryType and BookType are generated out of the schema and represent the complete document, the type of the <Library> element, and the complex type BookType, which is the type of the <Book> element. Note that the automatic name de-collision renamed the Library class to Library2 to avoid a possible conflict with the library namespace name.

```
protected static void example() throws Exception {
        // create a new, empty XML document
        Library2 libDoc = Library2.createDocument();

        // create the root element <Library> and add it to the document
        LibraryType lib = libDoc.Library3.append();

        // create a new <Book> and add it to the library
        BookType book = lib.Book.append();

        // set the ISBN attribute of the book
        book.ISBN.setValue("0764549642");

        // add the <Title> and <Author> elements, and set values
        book.Title.append().setValue("The XML Spy Handbook");
        book.Author.append().setValue("Altova");

        // set the schema location (this is optional)
        libDoc.setSchemaLocation("Library.xsd");

        // save the XML document to a file with default encoding (UTF-8)
. "true" causes the file to be pretty-printed.
        libDoc.saveToFile("Library1.xml", true);
}
```

Note that all elements are created by calling append(), and that attributes (like ISBN in the example) can simply be accessed like structure members.

Reading XML Documents

The following example reads the file we just created and demonstrates the various methods for reading XML files:

```
protected static void example() throws Exception {
        // load XML document
        Library2 libDoc = Library2.loadFromFile("Library1.xml");
```

```
                // get the first (and only) root element <Library>
                LibraryType lib = libDoc.Library3.first();

                // it is possible to check whether an element exists:
                if (!lib.Book.exists())
                {
                        System.out.println("This library is empty.");
                        return;
                }

                // iteration: for each <Book>...
                for (java.util.Iterator itBook = lib.Book.iterator();
itBook.hasNext(); )
                {
                        BookType book = (BookType) itBook.next();

                        // output values of ISBN attribute and (first and only)
title element
                        System.out.println("ISBN: " + book.ISBN.getValue());
                        System.out.println("Title: " + book.Title.first().get
Value());

                        // for each <Author>...
                        for (java.util.Iterator itAuthor = book.Author.iterator();
itAuthor.hasNext(); )
                                System.out.println("Author: " +
((com.Library.xs.stringType)itAuthor.next()).getValue());

                        // alternative: use count and index
                        for (int j = 0; j < book.Author.count(); ++j)
                                System.out.println("Author: " + book.Author.at(j)
.getValue());
                }
        }
```

Note the type of the book.Author member: xs.stringType is a generated class for any element with simple string content. Similar classes are generated for all other XML Schema types.

Of course you can also load a document, modify it by adding or deleting elements and attributes, and then save the changed file back to disk.

Error Handling

Errors are reported by exceptions. The following exception classes are defined in the namespace com.altova:

Class	Base Class	Description
SourceInstanceUnavailableException	Exception	A problem occurred while loading an XML instance.
TargetInstanceUnavailableException	Exception	A problem occurred while saving an XML instance.

In addition, the following Java exceptions are commonly used:

Class	Description
java.lang.Error	Internal program logic error (independent of input data)
java.lang.Exception	Base class for runtime errors
java.lang.IllegalArgumentsException	A method was called with invalid argument values, or a type conversion failed.

 © 2007 Altova GmbH

java.lang.ArithmeticException	Exception thrown when a numeric type conversion fails.

14.4.3 Using the generated C++ library

Please note: This topic describes code generation when using the option **Generate code compatible to V2007r3** in the code generation dialog.

Generated Libraries

Name	Purpose
Altova	Base library containing common runtime support, identical for every schema
AltovaXML	Base library containing runtime support for XML (MSXML or Xerces), identical for every schema
YourSchema	Library containing declarations generated from the input schema, named as the schema file or DTD
YourSchema Test	Test application skeleton (XMLSpy only)

DOM Implementations
The generated C++ code supports either Microsoft MSXML or Apache Xerces 2.6 or higher, depending on code generation settings. The syntax for using the generated code is identical for both DOM implementations.

Character Types
The generated C++ code can be compiled with or without Unicode support. Depending on this setting, the type **tstring** will be defined to **std::string** or **std::wstring**, consisting of narrow or wide characters. To use Unicode characters in your XML file that are not representable with the current 8-bit character set, Unicode support must be enabled. Please take care with the **_T()** macros. This macro ensures that string constants are stored correctly, whether you're compiling for Unicode or non-Unicode programs.

Data Types
The default mapping of XML Schema types to C++ data types is:

XML Schema	C++	Remarks
xs:string	string_type	string_type is defined as std::string or std:wstring
xs:boolean	bool	
xs:decimal	double	C++ does not have a decimal type, so double is used.
xs:float, xs:double	double	
xs:integer	__int64	xs:integer has unlimited range, mapped to __int64 for efficiency reasons.
xs:nonNegativeInteger	unsigned __int64	see above
xs:int	int	
xs:unsignedInt	unsigned int	
xs:dateTime, date, time, gYearMonth, gYear, gMonthDay, gDay, gMonth	altova::DateTime	
xs:duration	altova::Duration	
xs:hexBinary and xs:base64Binary	std::vector<unsigned char>	Encoding and decoding of binary data is done automatically.

xs:anySimpleType	string_type	

All XML Schema types not contained in this list are derived types, and mapped to the same C++ type as their respective base type.

Generated Document Class

In addition to the classes for the types declared in the XML Schema, the document class is generated. It contains all possible root elements as members, and in addition the following methods ("*CDoc*" stands for the name of the generated document class itself):

Method	Purpose
static *CDoc* **LoadFromFile**(const string_type& fileName)	Loads an XML document from a file
static *CDoc* **LoadFromString**(const string_type& xml)	Loads an XML document from a string
static *CDoc* **LoadFromBinary**(const std:vector<unsigned char>& xml)	Loads an XML document from a byte array
void **SaveToFile**(const string_type& fileName, bool prettyPrint, const string_type& encoding = "UTF-8")	Saves an XML document to a file with the specified encoding. If no encoding is specified, UTF-8 is assumed.
string_type **SaveToString**(bool prettyPrint)	Saves an XML document to a string
std::vector<unsigned char> **SaveToBinary** (bool prettyPrint, const string_type& encoding = "UTF-8")	Saves an XML document to a byte array with the specified encoding. If no encoding is specified, UTF-8 is assumed.
static *CDoc* **CreateDocument**()	Creates a new, empty XML document. Must be released using DestroyDocument().
void **DestroyDocument**()	Destroys a document. All references to the document and its nodes are invalidated. This must be called when finished working with a document.
void **SetSchemaLocation**(const string_type& schemaLocation)	Adds an xsi:schemaLocation or xsi:noNamespaceSchemaLocation attribute to the root element. A root element must already exist.
void **SetDTDLocation**(const string_type& dtdLocation)	Adds a DOCTYPE declaration with the specified system ID. A root element must already exist. This method is not supported for MSXML, since it is not possible to add a DOCTYPE declaration to a document in memory.

Generated Member Attribute Class

For each member attribute of a type, a class with the following methods is created.

Method	Purpose
bool **exists**()	Returns true if the attribute exists
void **remove**()	Removes the attribute from its parent element

In addition, assignment and conversion operators from/to the attribute's native type are created, so it can be used directly in assignments.

Generated Member Element Class

 © 2007 Altova GmbH

For each member element of a type, a class with the following methods is created.

Method	Purpose
MemberType **first**()	Returns the first instance of the member element
MemberType operator**[]**(unsigned int index)	Returns the member element specified by the index
MemberType **last**()	Returns the last instance of the member element
MemberType **append**()	Creates a new element and appends it to its parent
bool **exists**()	Returns true if at least one element exists
unsigned int **count**()	Returns the count of elements
void **remove**()	Deletes all occurrences of the element from its parent
Iterator<*MemberType*> **all**()	Returns an object for iterating instances of the member element

For simple and mixed content elements, assignment and conversion operators from/to the element's native type are created, so it can be used directly in assignments.

Creating XML Documents

The generated code library makes it very easy to create XML files. Let's start with an example XML file for the example schema:

```
<?xml version="1.0" encoding="UTF-8"?>
<Library xsi:schemaLocation="http://www.nanonull.com/LibrarySample Library.xsd" xmlns="
http://www.nanonull.com/LibrarySample" xmlns:xsi="
http://www.w3.org/2001/XMLSchema-instance">
    <Book ISBN="0764549642">
            <Title>The XML Spy Handbook</Title>
            <Author>Altova</Author>
    </Book>
</Library>
```

The following C++ code was used to create this file. The classes CLibrary, CLibraryType and CBookType are generated out of the schema and represent the complete document, the type of the <Library> element, and the complex type BookType, which is the type of the <Book> element.

```
using namespace Library;
void Example()
{
    // create a new, empty XML document
    CLibrary libDoc = CLibrary::CreateDocument();

    // create the root element <Library> and add it to the document
    CLibraryType lib = libDoc.Library2.append();

    // create a new <Book> and add it to the library
    CBookType book = lib.Book.append();

    // set the ISBN attribute of the book
    book.ISBN = _T("0764549642");

    // add the <Title> and <Author> elements, and set values
    book.Title.append() = _T("The XML Spy Handbook");
    book.Author.append() = _T("Altova");
```

```
    // set the schema location (this is optional)
    libDoc.SetSchemaLocation(_T("Library.xsd"));

    // save the XML document to a file with default encoding (UTF-8).
"true" causes the file to be pretty-printed.
    libDoc.SaveToFile(_T("Library1.xml"), true);

    // destroy the document - all references to the document and any of its
nodes become invalid
    libDoc.DestroyDocument();
}
```

Note that all elements are created by calling append(), and that attributes (like ISBN in the example) can simply be accessed like structure members. Remember to destroy the document when you are finished using it.

Reading XML Documents

The following example reads the file we just created and demonstrates the various methods for reading XML files:

```
using namespace Library;
void Example()
{
    // load XML document
    CLibrary libDoc = CLibrary::LoadFromFile(_T("Library1.xml"));

    // get the first (and only) root element <Library>
    CLibraryType lib = libDoc.Library2.first();

    // it is possible to check whether an element exists:
    if (!lib.Book.exists())
    {
           tcout << "This library is empty." << std::endl;
           return;
    }

    // iteration: for each <Book>...
    for (Iterator<CBookType> itBook = lib.Book.all(); itBook; ++itBook)
    {
           // output values of ISBN attribute and (first and only) title
element
           tcout << "ISBN: " << tstring(itBook->ISBN) << std::endl;
           tcout << "Title: " << tstring(itBook->Title.first()) << std::
endl;

           // for each <Author>...
           for (CBookType::Author::iterator itAuthor = itBook->Author.all
(); itAuthor; ++itAuthor)
                  tcout << "Author: " << tstring(itAuthor) << std::endl;

           // alternative: use count and index
           for (unsigned int j = 0; j < itBook->Author.count(); ++j)
                  tcout << "Author: " << tstring(itBook->Author[j]) << std::
endl;
    }

    // destroy the document - all references to the document and any of its
nodes become invalid
    libDoc.DestroyDocument();
}
```

To access the contents of attributes and elements as strings for the cout << operator, an explicit

 © 2007 Altova GmbH

cast to tstring is required.
Of course you can also load a document, modify it by adding or deleting elements and attributes, and then save the changed file back to disk.

Error Handling
Errors are reported by exceptions. The following exception classes are defined in the namespace altova:

Class	Base Class	Description
Error	std::logic_error	Internal program logic error (independent of input data)
Exception	std::runtime_error	Base class for runtime errors
InvalidArgumentsException	Exception	A method was called with invalid argument values.
ConversionException	Exception	Exception thrown when a type conversion fails
StringParseException	ConversionException	A value in the lexical space cannot be converted to value space.
ValueNotRepresentableException	ConversionException	A value in the value space cannot be converted to lexical space.
OutOfRangeException	ConversionException	A source value cannot be represented in target domain.
InvalidOperationException	Exception	An operation was attempted that is not valid in the given context.
DataSourceUnavailableException	Exception	A problem occurred while loading an XML instance.
DataTargetUnavailableException	Exception	A problem occurred while saving an XML instance.

All exception classes contain a message text and a pointer to a possible inner exception.

Method	Purpose
string_type **message**()	Returns a textual description of the exception
std::exception **inner**()	Returns the exception that caused this exception, if available, or NULL

14.4.4 Using the generated C# library

Please note: This topic describes code generation when using the option **Generate code compatible to V2007r3** in the code generation dialog.

Generated Libraries

Name	Purpose
Altova	Base library containing common runtime support, identical for every schema
AltovaXML	Base library containing runtime support for XML, identical for every schema
YourSchema	Library containing declarations generated from the input schema, named as the schema file or DTD
YourSchema Test	Test application skeleton (XMLSpy only)

DOM Implementation
The generated C# code uses the .NET standard System.Xml library as the underlying DOM

implementation.

Data Types
The default mapping of XML Schema types to C# data types is:

XML Schema	C#	Remarks
xs:string	string	
xs:boolean	bool	
xs:decimal	decimal	xs:decimal has unlimited range and precision, mapped to decimal for efficiency reasons.
xs:float, xs:double	double	
xs:long	long	
xs:unsignedLong	ulong	
xs:int	int	
xs:unsignedInt	uint	
xs:dateTime, date, time, gYearMonth, gYear, gMonthDay, gDay, gMonth	Altova.Types.DateTime	
xs:duration	Altova.Types.Duration	
xs:hexBinary and xs:base64Binary	byte[]	Encoding and decoding of binary data is done automatically.
xs:anySimpleType	string	

All XML Schema types not contained in this list are derived types, and mapped to the same C# type as their respective base type.

Generated Document Class
In addition to the classes for the types declared in the XML Schema, the document class is generated. It contains all possible root elements as members, and in addition the following methods ("Doc" stands for the name of the generated document class itself):

Method	Purpose
static *Doc* **LoadFromFile**(string fileName)	Loads an XML document from a file
static *Doc* **LoadFromString**(string xml)	Loads an XML document from a string
static *Doc* **LoadFromBinary**(byte[] xml)	Loads an XML document from a byte array
void **SaveToFile**(string fileName, bool prettyPrint, string encoding = "UTF-8")	Saves an XML document to a file with the specified encoding. If no encoding is specified, UTF-8 is assumed.
string **SaveToString**(bool prettyPrint)	Saves an XML document to a string
byte[] **SaveToBinary**(bool prettyPrint, string encoding = "UTF-8")	Saves an XML document to a byte array with the specified encoding. If no encoding is specified, UTF-8 is assumed.
static *Doc* **CreateDocument**()	Creates a new, empty XML document
void **SetSchemaLocation**(string schemaLocation)	Adds an xsi:schemaLocation or xsi:noNamespaceSchemaLocation attribute to the root element. A root element must already exist.
void **SetDTDLocation**(string dtdLocation)	Adds a DOCTYPE declaration with the specified system ID. A root element must already exist.

© 2007 Altova GmbH

Generated Member Attribute Class
For each member attribute of a type, a class with the following methods or properties is created.

Method	Purpose
bool **Exists**()	Returns true if the attribute exists
void **Remove**()	Removes the attribute from its parent element
AttributeType **Value**	Sets or gets the attribute value

Generated Member Element Class
For each member element of a type, a class with the following methods or properties is created. The class implements the standard System.Collections.IEnumerable interface, so it can be used with the *foreach* statement.

Method	Purpose
MemberType **First**	Returns the first instance of the member element
MemberType this**[**int index**]**	Returns the member element specified by the index
MemberType **At**(int index)	Returns the member element specified by the index
MemberType **Last**	Returns the last instance of the member element
MemberType **Append**()	Creates a new element and appends it to its parent
bool **Exists**	Returns true if at least one element exists
int **Count**	Returns the count of elements
void **Remove**()	Deletes all occurrences of the element from its parent
System.Collections.IEnumerator **GetEnumerator**()	Returns an object for iterating instances of the member element.
MemberType **Value**	Sets or gets the element content (only generated if element can have mixed or simple content)

Creating XML Documents
The generated code library makes it very easy to create XML files. Let's start with an example XML file for the example schema:

```
<?xml version="1.0" encoding="UTF-8"?>
<Library xsi:schemaLocation="http://www.nanonull.com/LibrarySample Library.xsd" xmlns="
http://www.nanonull.com/LibrarySample" xmlns:xsi="
http://www.w3.org/2001/XMLSchema-instance">
    <Book ISBN="0764549642">
        <Title>The XML Spy Handbook</Title>
        <Author>Altova</Author>
    </Book>
</Library>
```

The following C# code was used to create this file. The classes Library2, LibraryType and BookType are generated out of the schema and represent the complete document, the type of the <Library> element, and the complex type BookType, which is the type of the <Book> element. Note that the automatic name de-collision renamed the Library class to Library2 to avoid a possible conflict with the library namespace name.

```
using Library;
...
    protected static void Example()
    {
            // create a new, empty XML document
            Library2 libDoc = Library2.CreateDocument();

            // create the root element <Library> and add it to the document
            LibraryType lib = libDoc.Library3.Append();

            // create a new <Book> and add it to the library
            BookType book = lib.Book.Append();

            // set the ISBN attribute of the book
            book.ISBN.Value = "0764549642";

            // add the <Title> and <Author> elements, and set values
            book.Title.Append().Value = "The XML Spy Handbook";
            book.Author.Append().Value = "Altova";

            // set the schema location (this is optional)
            libDoc.SetSchemaLocation("Library.xsd");

            // save the XML document to a file with default encoding
(UTF-8). "true" causes the file to be pretty-printed.
            libDoc.SaveToFile("Library1.xml", true);
    }
```

Note that all elements are created by calling Append(), and that attributes (like ISBN in the example) can simply be accessed like structure members.

Reading XML Documents

The following example reads the file we just created and demonstrates the various methods for reading XML files:

```
using Library;
...
    protected static void Example()
    {
            // load XML document
            Library2 libDoc = Library2.LoadFromFile("Library1.xml");

            // get the first (and only) root element <Library>
            LibraryType lib = libDoc.Library3.First;

            // it is possible to check whether an element exists:
            if (!lib.Book.Exists)
            {
                    Console.WriteLine("This library is empty.");
                    return;
            }

            // iteration: for each <Book>...
            foreach (BookType book in lib.Book)
            {
                    // output values of ISBN attribute and (first and only)
title element
                    Console.WriteLine("ISBN: " + book.ISBN.Value);
                    Console.WriteLine("Title: " + book.Title.First.Value);

                    // for each <Author>...
                    foreach (xs.stringType author in book.Author)
                            Console.WriteLine("Author: " + author.Value);
```

 © 2007 Altova GmbH

```
                        // alternative: use count and index
                        for (int j = 0; j < book.Author.Count; ++j)
                                Console.WriteLine("Author: " + book.Author[j
].Value);
                }
        }
```

Note the type of the book.Author member: xs.stringType is a generated class for any element with simple string content. Similar classes are generated for all other XML Schema types.

Of course you can also load a document, modify it by adding or deleting elements and attributes, and then save the changed file back to disk.

Error Handling
Errors are reported by exceptions. The following exception classes are defined in the namespace Altova:

Class	Base Class	Description
ConversionException	Exception	Exception thrown when a type conversion fails
StringParseException	ConversionException	A value in the lexical space cannot be converted to value space.
DataSourceUnavailableException	System.Exception	A problem occurred while loading an XML instance.
DataTargetUnavailableException	System.Exception	A problem occurred while saving an XML instance.

In addition, the following .Net exceptions are commonly used:

Class	Description
System.Exception	Base class for runtime errors
System.ArgumentException	A method was called with invalid argument values, or a type conversion failed.
System.FormatException	A value in the lexical space cannot be converted to value space.
System.InvalidCastException	A value cannot be converted to another type.
System.OverflowException	A source value cannot be represented in target domain.

14.4.5 Using generated code compatible to old versions

Compatibility Mode
The code generator has been improved multiple times in the recent releases of Altova products, sometimes in ways that require changes to the code that uses the generated libraries. To preserve compatibility with existing code using libraries generated with earlier releases of XMLSpy, you can set the compatibility mode to the desired release. Default is version 2007r3, which creates the latest version of the code libraries described in the previous chapters.

The following chapters describe usage of the generated classes when using one of the following compatibility options:

- Code compatible to version 2005r3
- Code compatible to version 2006 to 2007

Please note that these compatibility options will be removed in a future version.

Version 2005r3:
Code generated code for XML schemas up till V2005R3 uses a static DOM document instance as parent for all nodes. This allows creating free-standing nodes that can be added to a document later, but may rise issues with multi-threading or memory management.

```
{
  LibraryType lib=new LibraryType();
   BookType book = new BookType();
   book.addISBN(new SchemaString("0764549642"));
   book.addTitle(new SchemaString("The XMLSpy Handbook"));
   book.addAuthor(new SchemaString("Altova"));
   lib.addBook(book);

   LibraryDoc doc=new LibraryDoc();
   doc.setRootElementName("http://www.nanonull.com/LibrarySample","Library");
   doc.setSchemaLocation("Library.xsd");//optional
   doc.save("Library1.xml",lib);
}
```

Version 2006 to 2007:
In C# and Java code generated with version 2006 to 2007, the standard (non-parameter) constructor is not used, because it does not establish a reference to a DOM document. A new document is created first which can then be referenced by the root node. All new child nodes must then also be created in the correct document. This is accomplished by using the generated factory functions called newXXX, please see the example code below.

```
{
  LibraryDoc doc=new LibraryDoc();
  doc.setSchemaLocation("Library.xsd");//optional

    // create root element (with no namespace prefix)
    LibraryType lib = new LibraryType(doc, "
http://www.nanonull.com/LibrarySample", "", "Library" );

    BookType book = lib.newBookType();    // factory functions are
generated for all members of a complex type
    book.addISBN( new SchemaString( "0764549642" ) );
    book.addTitle( new SchemaString( "The XML Spy Handbook" ) );
    book.addAuthor( new SchemaString( "Altova" ) );
    lib.addBook( book );

    doc.save( "Library1.xml", lib );
}
```

Version 2007r3:
The code generated by version 2007r3 has been completely redesigned to allow for multi-threading, avoid name collisions, and simplify the usage of the libraries. For details, see Using the generated code library.

Creating XML files (XMLSpy 2006)

Please note: This topic describes code generation when using the option **Generate code compatible to V2006-2007** in the code generation dialog.

Java:

To give a general overview, here is the code to create a test file:

```
protected static void example() throws Exception {
        LibraryDoc doc = new LibraryDoc();
        doc.setSchemaLocation("Library.xsd");  // optional
        LibraryType lib = new LibraryType(doc,
"http://www.nanonull.com/LibrarySample", "", "Library" );
        BookType book = lib.newBook();
        book.addISBN( new SchemaString( "0764549642" ) );
        book.addTitle( new SchemaString( "The XML Spy Handbook" ) );
        book.addAuthor( new SchemaString( "Altova" ) );
        lib.addBook( book );
        doc.save("Library1.xml", lib);
}
```

The idea of the generated classes is the following:
Create a document and associate it with the root element of the generated schema so that the tree is stored in the document.

First we create a new document:

```
LibraryDoc doc = new LibraryDoc();
```

Optionally we can also set the schema location.

```
doc.setSchemaLocation( "Library.xsd" );
```

We create the root element, passing to the method the document object that we have already generated, the namespace of `library.xsd`, and the name of the root element in the schema:

```
LibraryType lib = new LibraryType(doc,
"http://www.nanonull.com/LibrarySample", "", "Library" );
```

LibraryType is a class that was generated. It is similar to the <xs:element name="Library"> in the schema file. This is the root element for the xml files being generated (as in generate sample xml file).

We now have a new Library object. All we have to do is fill it with Book objects. Look at the schema again and notice that books are stored as a sequence within the Library.

The next step is to make an object of **BookType** and fill the new object.

```
BookType book = lib.newBook();
```

Definition of BookType in the schema file:

```
<xs:complexType name="BookType">
    <xs:sequence>
        <xs:element name="Title" type="xs:string"/>
        <xs:element name="Author" type="xs:string" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="ISBN" type="xs:string" use="required"/>
</xs:complexType>
```

We will insert an ISBN number, a title and an author.

```
book.addISBN( new SchemaString( "0764549642" ) );
book.addTitle( new SchemaString( "The XML Spy Handbook" ) );
```

```
book.addAuthor( new SchemaString( "Altova" ) );
```

We filled the BookType object, and we now have to add it to the library sequence:

```
lib.addBook(book);
```

Finally, save the file. The first parameter is the file name, the second is the root element itself.

```
doc.save( "Library1.xml", lib );
```

C++:

The C++ implementation is very similar to the Java one, we will therefore only discuss the differences between the two.

The function to be inserted above the main function is as follows:

```
void Example()
{
	CLibraryDoc doc;
	CLibraryType lib;
	doc.SetRootElementName(_T("http://www.nanonull.com/LibrarySample"), _T(
"Library"));
	doc.SetSchemaLocation(_T("Library.xsd")); // optional
	CBookType book;
	book.AddISBN( _T("0764549642") );
	book.AddTitle( _T("The XML Spy Handbook") );
	book.AddAuthor( _T("Altova") );
	lib.AddBook( book );
	doc.Save(_T("Library1.xml"), lib);
}
```

First off, you'll notice that all classes begin with a capital C - the rest of the class naming stays the same. All member function names start with a capital letter.

Please take care with the **_T** macros. The macro ensures that the string constant is stored correctly, whether you're compiling for Unicode or non-Unicode programs.

C#:

```
protected static void Example()
{
	LibraryDoc doc = new LibraryDoc();
	doc.SetSchemaLocation( "Library.xsd" ); // optional
	// create root element with no namespace prefix
	LibraryType lib = new LibraryType(doc, "
http://www.nanonull.com/LibrarySample", "", "Library" );
	BookType book = lib.NewBook();		// factory functions are generated for all
members of a complex type
	book.AddISBN( new SchemaString( "0764549642" ) );
	book.AddTitle( new SchemaString( "The XML Spy Handbook" ) );
	book.AddAuthor( new SchemaString( "Altova" ) );
	lib.AddBook( book );
	doc.Save( "Library1.xml", lib );
}
```


© 2007 Altova GmbH

Creating XML files (XMLSpy 2005)

Please note: This topic describes code generation when using the option **Generate code compatible to V2005R3** in the code generation dialog.

Java:

To give a general overview, here is the code to create a test file:

```
protected static void example() {
        LibraryType lib = new LibraryType();
        BookType book = new BookType();
        book.addISBN( new SchemaString( "0764549642" ) );
        book.addTitle( new SchemaString( "The XML Spy Handbook" ) );
        book.addAuthor( new SchemaString( "Altova" ) );
        lib.addBook( book );

        LibraryDoc doc = new LibraryDoc();
        doc.setRootElementName( "http://www.nanonull.com/LibrarySample", "Library"
);
        doc.setSchemaLocation( "Library.xsd" ); // optional
        doc.save( "Library1.xml", lib );
}
```

The idea of the generated classes is the following:
Generate the tree you want to save in the new file, create a document and store the tree in it.

The first line:

```
LibraryType lib = new LibraryType();
```

LibraryType is a class that was generated. It is similar to the <xs:element name="Library"> in the schema file. This is the root element for the xml files being generated (as in generate sample xml file).

We now have a new Library object. All we have to do is fill it with Book objects. Look at the schema again and notice that books are stored as a sequence within the Library.

The next step is to make an object of **BookType** and fill the new object.

```
BookType book = new BookType();
```

Definition of BookType in the schema file:

```
<xs:complexType name="BookType">
    <xs:sequence>
        <xs:element name="Title" type="xs:string"/>
        <xs:element name="Author" type="xs:string" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="ISBN" type="xs:string" use="required"/>
</xs:complexType>
```

We will insert an ISBN number, a title and an author.

```
book.addISBN( new SchemaString( "0764549642" ) );
book.addTitle( new SchemaString( "The XML Spy Handbook" ) );
book.addAuthor( new SchemaString( "Altova" ) );
```

We filled the BookType object, and we now have to add it to the sequence of **internationalPriceTypes** within Test_Empty:

```
lib.addBook(book);
```

Done, the tree is complete! All we have to do now, is create a new document and define the namespace and name of the root element:

```
LibraryDoc doc = new LibraryDoc();
doc.setRootElementName( "http://www.nanonull.com/LibrarySample", "Library"
);
```

Optionally we can also set the schema location.

```
doc.setSchemaLocation( "Library.xsd" );
```

Finally, save the file, - the first parameter is the file name, the second is the root element itself.

```
doc.save( "Library1.xml", lib );
```

C++:

The C++ implementation is very similar to the Java one, we will therefore only discuss the differences between the two.

The function to be inserted above the main function is as follows:

```
void Example()
{
	CLibraryType lib;
	CBookType book;
	book.AddISBN( _T("0764549642") );
	book.AddTitle( _T("The XML Spy Handbook") );
	book.AddAuthor( _T("Altova") );
	lib.AddBook( book );

	CLibraryDoc doc;
	doc.SetRootElementName( _T("http://www.nanonull.com/LibrarySample"), _T(
"Library") );
	doc.SetSchemaLocation( _T("Library.xsd") ); // optional
	doc.Save( _T("Library1.xml"), lib );
}
```

First off, you'll notice that all classes begin with a capital C - the rest of the class naming stays the same. All member function names start with a capital letter.

Please take care with the **_T** macros. The macro ensures that the string constant, is stored correctly, whether you're compiling for Unicode or non Unicode programs.

C#:

```
protected static void Example()
{
	LibraryType lib = new LibraryType();
	BookType book = new BookType();
	book.AddISBN( new SchemaString( "0764549642" ) );
```

 © 2007 Altova GmbH

```
        book.AddTitle( new SchemaString( "The XML Spy Handbook" ) );
        book.AddAuthor( new SchemaString( "Altova" ) );
        lib.AddBook( book );

        LibraryDoc doc = new LibraryDoc();
        doc.SetRootElementName( "http://www.nanonull.com/LibrarySample", "Library"
);
        doc.SetSchemaLocation( "Library.xsd" ); // optional
        doc.Save( "Library1.xml", lib );
}
```

Opening and parsing existing XML files (XMLSpy 2006)

Please note: This topic describes code generation when using the option **Generate code compatible to V2006-2007** in the code generation dialog.

We will now use the **Library1.xml** file generated with our sample program. Please make sure that this file exists, or change the path to it in the load command.

Java:

Sample source code:

```
protected static void example2() {

        LibraryDoc doc = new LibraryDoc();
        LibraryType lib = new LibraryType(doc.load("Library1.xml"));

        if (!lib.hasBook()) {
                System.out.println("This library is empty");
                return;
        }

        for (int i = 0; i < lib.getBookCount(); i++) {
                BookType book;
                try {
                        book = lib.getBookAt(i);

                        System.out.println("ISBN:   " + book.getISBN().toString());
                        System.out.println("Title:  " + book.getTitle().toString());

                        for (int j = 0; j < book.getAuthorCount(); j++) {
                                System.out.println("Author: "+
book.getAuthorAt(j).toString());
                        }
                } catch (Exception e) {
                        e.printStackTrace();
                }
        }
}
```

in the ***main()*** function add a call to:

```
        example2();
```

after

```
        example1();
```

The rest is reasonably straightforward.
First create a new variable of type Document, and one for the root element, which is immediately filled by the **doc.load** function. The only parameter it needs, is the path to the xml file being used.

```
LibraryDoc doc = new LibraryDoc();
LibraryType lib = new LibraryType( doc.load( "Library1.xml" ) );
```

We now have a document and know that lib contains a sequence of BookType objects.

The next step is to dump the values of all of the Book elements:

```
if( !lib.hasBook() ) {
```

Why check for objects? The call just shows that a function exists that checks this for you.

Step through the books and output their values:

```
for( int i = 0; i < lib.getBookCount(); i++ ) {
        BookType book = lib.getBookAt( i );

        System.out.println( "ISBN:   " + book.getISBN().asString() );
        System.out.println( "Title:  " + book.getTitle().asString() );

        for( int j = 0; j < book.getAuthorCount(); j++ ) {
                System.out.println( "Author: " + book.getAuthorAt( j ).asString()
);
        }
}
```

We're finished, all the values are listed in your output window.

C++:

This section only explains the differences to the java code.

```
void Example2()
{
        CLibraryDoc doc;
        CLibraryType lib = doc.Load( _T("Library1.xml") );

        if ( !lib.HasBook() )
        {
        cout << "This library is empty" << endl;
        return;
        }

        for ( int i = 0; i < lib.GetBookCount(); i++ )
        {
                CBookType book = lib.GetBookAt( i );
                cout << "ISBN:   " << book.GetISBN() << endl;
                cout << "Title:  " << book.GetTitle() << endl;

                for ( int j = 0; j < book.GetAuthorCount(); j++ )
                {
                        cout << "Author: " << book.GetAuthorAt( j ) << endl;
                }
        }
```


© 2007 Altova GmbH

```
}
```

All member functions start with a capital letter, and all class names start with a C.

C#:

This section only explains the differences to the java code.

```
protected static void Example2()
{
	LibraryDoc doc = new LibraryDoc();
	LibraryType lib = new LibraryType( doc.Load( "Library1.xml" ) );

	if ( !lib.HasBook() )
	{
		Console.WriteLine( "This library is empty" );
		return;
	}

	for ( int i = 0; i < lib.GetBookCount(); i++ )
	{
		BookType book = lib.GetBookAt( i );
		Console.WriteLine( "ISBN:   {0}", book.GetISBN() );
		Console.WriteLine( "Title:  {0}", book.GetTitle() );

		for( int j = 0; j < book.GetAuthorCount(); j++ )
		{
			Console.WriteLine( "Author: {0}", book.GetAuthorAt( j ) );
		}
	}
}
```

All member functions start with a capital letter.

Opening and parsing existing XML files (XMLSpy 2005)

Please note: This topic describes code generation when using the option **Generate code compatible to V2005R3** in the code generation dialog.

We will now use the **Library1.xml** file generated with our sample program. Please make sure that this file exists, or change the path to it in the load command.

Java:

Sample source code:

```
protected static void example2() {
	LibraryDoc doc = new LibraryDoc();
	LibraryType lib = new LibraryType( doc.load( "Library1.xml" ) );

if( !lib.hasBook() ) {
	System.out.println( "This library is empty" );
	return;
}
```

```
        for( int i = 0; i < lib.getBookCount(); i++ ) {
                BookType book = lib.getBookAt( i );
                System.out.println( "ISBN: " + book.getISBN().asString() );
                System.out.println( "Title: " + book.getTitle().asString() );

                        for( int j = 0; j < book.getAuthorCount(); j++ ) {
                                System.out.println( "Author: " + book.getAuthorAt( j
).asString());
                        }
                }
        }
```

in the ***main()*** function add a call to:

```
        example2();
```

after

```
        example1();
```

The rest is reasonably straightforward.
First create a new variable of type Document, and one for the root element, which is immediately filled by the **doc.load** function. The only parameter it needs, is the path to the xml file being used.

```
        LibraryDoc doc = new LibraryDoc();
        LibraryType lib = new LibraryType( doc.load( "Library1.xml" ) );
```

We now have a document and know that lib contains a sequence of BookType objects.

The next step is to dump the values of all of the Book elements:

```
        if( !lib.hasBook() ) {
```

Why check for objects? The call just shows that a function exists, that checks this for you.

Step through the books and dump their values:

```
        for( int i = 0; i < lib.getBookCount(); i++ ) {
                BookType book = lib.getBookAt( i );

                System.out.println( "ISBN:   " + book.getISBN().asString() );
                System.out.println( "Title:  " + book.getTitle().asString() );

                for( int j = 0; j < book.getAuthorCount(); j++ ) {
                        System.out.println( "Author: " + book.getAuthorAt( j ).asString()
);
                }
        }
```

We're finished, all the values are listed in your output window.

C++:

This section only explains the differences to the java code.

```
        void Example2()
        {
                CLibraryDoc doc;
```

 © 2007 Altova GmbH

```
        CLibraryType lib = doc.Load( _T("Library1.xml") );

        if( !lib.HasBook() )
        {
                cout << "This library is empty" << endl;
                return;
        }

        for( int i = 0; i < lib.GetBookCount(); i++ )
        {
                CBookType book = lib.GetBookAt( i );

                cout << "ISBN:   " << book.GetISBN() << endl;
                cout << "Title:  " << book.GetTitle() << endl;

                for( int j = 0; j < book.GetAuthorCount(); j++ )
                {
                        cout << "Author: " << book.GetAuthorAt( j ) << endl;
                }
        }
}
```

All member functions start with a capital letter, and all Classnames start with a C.

C#:

This section only explains the differences to the java code.

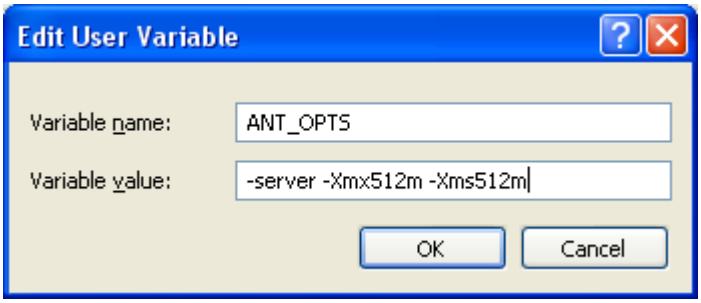
```
protected static void Example2()
{
        LibraryDoc doc = new LibraryDoc();
        LibraryType lib = new LibraryType( doc.Load( "Library1.xml" ) );

        if( !lib.HasBook() )
        {
                Console.WriteLine( "This library is empty" );
                return;
        }

        for( int i = 0; i < lib.GetBookCount(); i++ )
        {
                BookType book = lib.GetBookAt( i );

                Console.WriteLine( "ISBN:   {0}", book.GetISBN() );
                Console.WriteLine( "Title:  {0}", book.GetTitle() );

                for( int j = 0; j < book.GetAuthorCount(); j++ )
                {
                        Console.WriteLine( "Author: {0}", book.GetAuthorAt( j ) );
                }
        }
}
```

All member functions start with a capital letter.

14.5 Code generation tips

Out-of-memory exceptions during Java compilation and how to resolve them
Complex schemas can produce a large amount of code, which might cause a java.lang.OutofMemory exception during compilation using ANT. To rectify this:

- Add the environment variable **ANT_OPTS,** which sets specific ANT options such as the memory to be allocated to the compiler, and add values as shown below.



You may need to customize the values depending on the amount of memory in your machine and the size of the project you are working with. For more details see your Java VM documentation.

Reserving method names
When customizing code generation using the supplied spl files, it might be necessary to reserve names to avoid collisions with other symbols. To do this:

1. Navigate to subdirectory corresponding to the programming language of the **spl** subdirectory of the program installation directory e.g. C:\Program Files\Altova\XMLSpy2007\spl\java\.
2. Open the **settings.spl** file and insert a new line into the reserve section e.g. reserve "myReservedWord".
3. Regenerate the program code.

XML Schema support
The following XML Schema constructs are translated into code:

- XML Namespaces: Mapped to namespaces in the target programming language

- Simple types
  - Built-in XML Schema types: Mapped to native primitive types or classes in the Altova types library
  - Derived by extension
  - Derived by restriction
  - Facets
  - Enumerations
  - Patterns

- Complex types
  - Built-in anyType node
  - User-defined complex types
  - Derived by extension
  - Derived by restriction

© 2007 Altova GmbH

- Complex content
- Simple content
- Mixed content

- The following advanced XML Schema features are not, or not fully supported:
  - Content models (sequence, choice, all): Top-level compositor available in SPL, but not enforced by generated classes
  - default and fixed values for attributes: Available in SPL, but not set or enforced by generated classes
  - attributes xsi:type, abstract types: SetXsiType methods are generated and need to be called by the user
  - Union types: Not all combinations are supported
  - Substitution groups: Partial support
  - attribute nillable="true" and xsi:nil
  - uniqueness constraints
  - key and keyref

14.6 Code generator options

The menu option **DTD/Schema | Generate Program Code** lets you specify the target programming language as well as specific code generation settings.

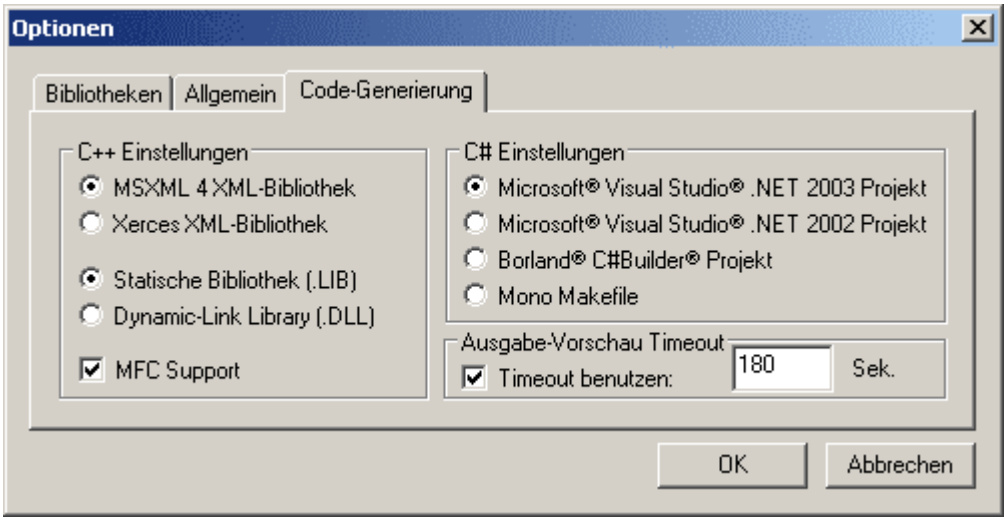
C++ Settings tab:
This tab lets you define the settings for C++ and C# programming languages.

C++

- Microsoft Visual Studio 2005, .NET 2003 or Visual C++ 6.0 project file
- generate code using either MSXML 4 or Apache Xerces XML library
- generate static libraries, or Dynamic-link libraries
- generate code with or without MFC support

C# Settings tab:
C#
Select the type of project file you want to generate:

- Microsoft Visual studio 2005, 2003, or 2002 project file
- Borland C#Builder project
- Mono makefile



Compatibility Mode:
The code generator has been improved multiple times in the recent releases of Altova products, sometimes in ways that require changes to the code that uses the generated libraries. To preserve compatibility with existing code using libraries generated with earlier releases of XMLSpy, you can set the compatibility mode to the desired release. Default is version 2007r3, which creates the latest version of the code libraries.

- Please make it a point to use the new code generation features i.e., set the version to the latest available.
- The availability of the Compatibility Mode is only temporary, it will be removed in a future version.

© 2007 Altova GmbH

14.7 The way to SPL (Spy Programming Language)

This section gives an overview of Spy Programming Language, the code generator's template language.

It is assumed that you have prior programming experience, and are familiar with operators, functions, variables and classes, as well as the basics of object-oriented programming - which is used heavily in SPL. You should also have detailed knowledge of XML Schema.

The templates used by XMLSpy are supplied in the **...**\XMLSpy2007**spl** folder. You can use these files as an aid to help you in developing your own templates.

How code generator works
Inputs to the code generator are the template files (.spl) and the object model provided by XMLSpy. The template files contain SPL instructions for creating files, reading information from the object model and performing calculations, interspersed with literal code fragments in the target programming language.

The template file is interpreted by the code generator and outputs **.cpp, .java, .cs** source code files, project files, or any other type of file depending on the template. The source code can then be compiled into an executable file that accesses XML data described by the schema file.

SPL files have access to a wide variety of information that is collated from the source schemas. Please note that an SPL file is not tied to a specific schema, but allows access to all schemas! Make sure you write your SPL files generically, avoid structures etc. which apply to specific schemas!

Example: Creating a new file in SPL:

```
[create "test.cpp"]
#include "stdafx.h"
[close]
```

This is a very basic SPL file. It creates a file named **test.cpp,** and places the include statement within it. The close command completes the template.

14.7.1 Basic SPL structure

An SPL file contains literal text to output, interspersed with code generator instructions.

Code generator instructions are enclosed in square brackets '**[**' and '**]**'.
Multiple statements can be included in a bracket pair. Additional statements have to be separated by a new line or a colon '**:**'.

Valid examples are:

```
[$x = 42
$x = $x + 1]
```

or

```
[$x = 42: $x = $x + 1]
```

Adding text to files
Text not enclosed by **[** and **]**, is written directly to the current output file. If there is no current output file, the text is ignored (see Using files how to create an output file).
To output literal square brackets, escape them with a backslash: **\[** and **\]**; to output a backslash use ****.

Comments
Comments inside an instruction block always begin with a **'** character, and terminate on the next line, or at a block close character **]**.

14.7.2 Declarations

The following statements are evaluated while parsing the SPL template file. They are **not** affected by flow control statements like conditions, loops or subroutines, and are always evaluated exactly once.

These keywords, like all keywords in SPL, are not case sensitive.

Remember that all of these declarations must be inside a block delimited by square brackets.

map *mapname key* **to** *value* [, *key* **to** *value*]...

This statement adds information to a map. See below for specific uses.

map schemanativetype *schematype* **to** *typespec*

The specified built-in XML Schema type will be mapped to the specified native type or class, using the specified formatter. This setting applies only to code generation for version 2007r3 and higher. Typespec is a native type or class name, followed by a comma, followed by the formatter class instance.

Example:
```
map schemanativetype "double" to "double,Altova::DoubleFormatter"
```

map type *schematype* **to** *classname*

The specified built-in XML Schema type will be mapped to the specified class. This setting applies only to code generation for version 2007 or lower.

Example:
```
map type "float" to "CSchemaFloat"
```

default *setting* **is** *value*

This statement allows you to affect how class and member names are derived from the XML Schema.
Note that the setting names are case sensitive.

Example:
```
default "InvalidCharReplacement" is "_"
```

Setting name	Explanation
ValidFirstCharSet	Allowed characters for starting an identifier
ValidCharSet	Allowed characters for other characters in an identifier
InvalidCharReplacement	The character that will replace all characters in names that are not in the ValidCharSet
AnonTypePrefix	Prefix for names of anonymous types*
AnonTypeSuffix	Suffix for names of anonymous types*
ClassNamePrefix	Prefix for generated class names
ClassNameSuffix	Suffix for generated class names
EnumerationPrefix	Prefix for symbolic constants declared for enumeration values

 © 2007 Altova GmbH

EnumerationUpperCase	"on" to convert the enumeration constant names to upper case
FallbackName	If a name consists only of characters that are not in ValidCharSet, use this one

* Names of anonymous types are built from AnonTypePrefix + element name + AnonTypeSuffix

reserve *word*

Adds the specified word to the list of reserved words. This ensures that it will never be generated as a class or member name.

Example:
```
reserve "while"
```

include *filename*

Example:
```
include "Module.cpp"
```

includes the specified file as SPL source. This allows you to split your template into multiple files for easier editing and handling.

14.7.3 Variables

Any non-trivial SPL file will require variables. Some variables are predefined by the code generator, and new variables may be created simply by assigning values to them.

The **$** character is used when **declaring** or **using** a variable, a variable name is always prefixed by **$**.
Variable names are **case sensitive**.

Variables types:

- integer - also used as boolean, where 0 is false and everything else is true
- string
- object - provided by XMLSpy
- iterator - see foreach statement

Variable types are declared by first assignment:

```
[$x = 0]
```
x is now an integer.

```
[$x = "teststring"]
```
x is now treated as a string.

Strings
String constants are always enclosed in double quotes, like in the example above. **\n** and **\t** inside double quotes are interpreted as newline and tab, **\"** is a literal double quote, and **** is a backslash. String constants can also span multiple lines.

String concatenation uses the **&** character:

```
[$BasePath = $outputpath & "/" & $JavaPackageDir]
```

Objects
Objects represent the information contained in the XML schema. Objects have **properties**, which can be accessed using the **.** operator. It is not possible to create new objects in SPL (they are predefined by the code generator, derived from the input schema), but it is possible to assign objects to variables.

Example:

```
class [=$class.Name]
```

This example outputs the word "class", followed by a space and the value of the **Name** property of the **$class** object.

14.7.4 Predefined variables

After a Schema file is analyzed by the code generator, the objects in the table below exist in the Template Engine.

Current object model (for code compatibility to version 2007r3 and later)

Name	Type	Description
$schematype	integer	1 for DTD, 2 for XML Schema
$TheLibrary	Library	The library derived from the XML Schema or DTD
$module	string	name of the source Schema without extension
$outputpath	string	The output path specified by the user, or the default output path

Old object model (for code compatibility up to version 2007)

Name	Type	Description
$schematype	integer	1 for DTD, 2 for XML Schema
$namespaces	Namespace collection	Collection of Namespace objects
$classes	Class collection	All the complex types, elements,… in a flat view
$module	string	name of the source Schema without extension
$outputpath	string	The output path specified by the user, or the default output path

For **C++** generation only:

Name	Type	Description
$domtype	integer	1 for MSXML, 2 for Xerces
$libtype	integer	1 for static LIB, 2 for DLL
$mfc	boolean	True if MFC support is enabled
$vc6project	boolean	True if Visual C++ 6.0 project files are to be generated
$VS2003Project	boolean	True if Visual C++ 2003 project files are to be generated
$VS2005Project	boolean	True if Visual C++ 2005 project files are to be generated

For **C#** generation only:

Name	Type	Description
$CreateVSProject	boolean	True if Visual Studio .NET project files are to be generated

© 2007 Altova GmbH

$CreateVS2003Project	boolean	True if Visual Studio 2003 project files are to be generated
$CreateVS2005Project	boolean	True if Visual Studio 2005 project files are to be generated
$CreateBorlandProject	boolean	True if Borland C#Builder project files are to be generated
$CreateMonoMakefile	boolean	True if a Mono makefile is to be generated

14.7.5 Creating output files

These statements are used to create output files from the code generation.

Remember that all of these statements must be inside a block delimited by square brackets.

create *filename*

creates a new file. The file has to be closed with the **close** statement. All following output is written to the specified file.

Example:

```
[create $outputpath & "/" & $JavaPackageDir & "/" & $application.Name &
".java"]
package [=$JavaPackageName];

public class [=$application.Name]Application {
...
}
[close]
```

close

closes the current output file.

=$*variable*

writes the value of the specified variable to the current output file.

Example:

```
[$x = 20+3]
The result of your calculation is [=$x] - so have a nice day!
```

- The file output will be:

```
The result of your calculation is 23 - so have a nice day!
```

write *string*

writes the string to the current output file.

Example:

```
[write "C" & $name]
```

This can also be written as:

```
C[=$name]
```

filecopy *source* **to** *target*

copies the source file to the target file, without any interpretation.

Example:

```
filecopy "java/mapforce/mapforce.png" to $outputpath & "/" & $JavaPackageDir &
"/mapforce.png"
```

14.7.6 Operators

Operators in SPL work like in most other programming languages.

List of SPL operators in descending precedence order:

.	Access object property
( )	Expression grouping
true	boolean constant "true"
false	boolean constant "false"
&	String concatenation
-	Sign for negative number
not	Logical negation
*	Multiply
/	Divide
%	Modulo
+	Add
-	Subtract
<=	Less than or equal
<	Less than
>=	Greater than or equal
>	Greater than
=	Equal
<>	Not equal
and	Logical conjunction (with short circuit evaluation)
or	Logical disjunction (with short circuit evaluation)
=	Assignment

14.7.7 Conditions

SPL allows you to use standard "if" statements. The syntax is as follows:

```
if condition
   statements
else
   statements
endif
```

or, without else:

```
if condition
   statements
endif
```

 © 2007 Altova GmbH

Please note that there are no round brackets enclosing the condition!
As in any other programming language, conditions are constructed with logical and comparison operators.

Example:
```
[if $namespace.ContainsPublicClasses and $namespace.Prefix <> ""]
   whatever you want ['inserts whatever you want, in the resulting file]
[endif]
```

Switch
SPL also contains a multiple choice statement.

Syntax:
```
switch $variable
   case X:
      statements
   case Y:
   case Z:
      statements
   default:
      statements
endswitch
```

The case labels must be constants or variables.

The switch statement in SPL does not fall through the cases (as in C), so there is no need for a "break" statement.

14.7.8 foreach

Collections and iterators

A collection contains multiple objects - like a ordinary array. Iterators solve the problem of storing and incrementing array indexes when accessing objects.

Syntax:
```
foreach iterator in collection
   statements
next
```

Example:
```
[foreach $class in $classes
   if not $class.IsInternal
      ]   class [=$class.Name];
[   endif
next]
```

The first line:
$classes is the **global object** of all generated types. It is a collection of single class objects.

Foreach steps through all the items in $classes, and executes the code following the instruction, up to the **next** statement, for each of them.

In each iteration, **$class** is assigned to the next class object. You simply work with the class object instead of using, classes[i]->Name(), as you would in C++.

All collection iterators have the following additional properties:

Index	The current index, starting with 0
IsFirst	true if the current object is the first of the collection (index is 0)

IsLast	true if the current object is the last of the collection
Current	The current object (this is implicit if not specified and can be left out)

Example:

```
[foreach $enum in $facet.Enumeration
   if not $enum.IsFirst
      ], [
   endif
   ]"[=$enum.Value]"[
next]
```

14.7.9 Subroutines

Code generator supports subroutines in the form of procedures or functions.

Features:

- By-value and by-reference passing of values
- Local/global parameters (local within subroutines)
- Local variables
- Recursive invocation (subroutines may call themselves)

Subroutine declaration

Subroutines

Syntax example:

```
Sub SimpleSub()
... lines of code
EndSub
```

- **Sub** is the keyword that denotes the procedure.
- **SimpleSub** is the name assigned to the subroutine.
- Round **parenthesis** can contain a parameter list.
- The code block of a subroutine starts immediately after the closing parameter parenthesis.
- **EndSub** denotes the end of the code block.

Please note:
Recursive or cascaded subroutine **declaration** is not permitted, i.e. a subroutine may not contain another subroutine.

Parameters

Parameters can also be passed by procedures using the following syntax:

- All parameters must be variables
- Variables must be prefixed by the **$** character
- Local variables are defined in a subroutine
- Global variables are declared explicitly, outside of subroutines
- Multiple parameters are separated by the comma character "**,**" within round parentheses
- Parameters can pass values

Parameters - passing values

 © 2007 Altova GmbH

Parameters can be passed in two ways, by value and by reference, using the keywords **ByVal** and **ByRef** respectively.

Syntax:

```
' define sub CompleteSub()
[ Sub CompleteSub( $param, ByVal $paramByValue, ByRef $paramByRef )
] ...
```

- **ByVal** specifies that the parameter is passed by value. Note that most objects can only be passed by reference.
- **ByRef** specifies that the parameter is passed by reference. This is the default if neither ByVal nor ByRef is specified.

Function return values

To return a value from a subroutine, use the **return** statement. Such a function can be called from within an expression.

Example:

```
' define a function
[ Sub MakeQualifiedName( ByVal $namespacePrefix, ByVal $localName )
if $namespacePrefix = ""
  return $localName
else
  return $namespacePrefix & ":" & $localName
endif
EndSub
]
```

Subroutine invocation

Use **call** to invoke a subroutine, followed by the procedure name and parameters, if any.

```
Call SimpleSub()
```

or,

```
Call CompleteSub( "FirstParameter", $ParamByValue, $ParamByRef )
```

Function invocation

To invoke a function (any subroutine that contains a **return** statement), simply use its name inside an expression. Do not use the **call** statement to call functions.
Example:

```
$QName = MakeQualifiedName($namespace, "entry")
```

Subroutine example

Highlighted example showing subroutine declaration and invocation.

A sample SPL file:

```
[create $outputpath & $module & "output.txt"

' define sub SimpleSub()
sub SimpleSub()
]SimpleSub() called
[endsub

' execute sub SimpleSub()
call SimpleSub()

$ParamByValue    = "Original Value"
]ParamByValue    = [=$ParamByValue]
[$ParamByRef     = "Original Value"
]ParamByRef      = [=$ParamByRef]

' define sub CompleteSub()
[sub CompleteSub( $param, byval $paramByValue, byref $paramByRef )
]CompleteSub called.
    param = [=$param]
    paramByValue = [=$paramByValue]
    paramByRef = [=$paramByRef]
[$ParamByRef = "Local Variable"
$paramByValue = "new value"
$paramByRef = "new value"
]   Set values inside sub
[$ParamByRef = "Local Variable"
$paramByValue = "new value"
$paramByRef = "new value"
]CompleteSub finished.
[endsub

' run sub CompleteSub()
call CompleteSub( "FirstParameter", $ParamByValue, $ParamByRef )
]
ParamByValue=[=$ParamByValue]
ParamByRef=[=$ParamByRef]
[
close
]
```

The same sample code:

```
[create $outputpath & $module & "output.txt"

' define sub SimpleSub()
Sub SimpleSub()
]SimpleSub() called
[endsub

' execute sub SimpleSub()
Call SimpleSub()

$ParamByValue       = "Original Value"
]ParamByValue       = [=$ParamByValue]
[$ParamByRef  = "Original Value"
]ParamByRef   = [=$ParamByRef]

' define sub CompleteSub()
```


© 2007 Altova GmbH

```
[Sub CompleteSub( $param, ByVal $paramByValue, ByRef $paramByRef )
]CompleteSub called.
       param = [=$param]
       paramByValue = [=$paramByValue]
       paramByRef = [=$paramByRef]
[$ParamByRef = "Local Variable"
$paramByValue = "new value"
$paramByRef = "new value"
]      Set values inside Sub
[$ParamByRef = "Local Variable"
$paramByValue = "new value"
$paramByRef = "new value"
]CompleteSub finished.
[endsub

' run sub CompleteSub()
Call CompleteSub( "FirstParameter", $ParamByValue, $ParamByRef )
]
ParamByValue=[=$ParamByValue]
ParamByRef=[=$ParamByRef]
[
Close
]
```

14.7.10 Built in Types

The section describes the properties of the built-in types used in the predefined variables which describe the parsed schema.

Library

This object represents the whole library generated from the XML Schema or DTD.

Property	Type	Description
SchemaNamespaces	Namespace collection	Namespaces in this library
SchemaFilename	string	Name of the XSD or DTD file this library is derived from
SchemaType	integer	1 for DTD, 2 for XML Schema
Guid	string	A globally unique ID
CodeName	string	Generated library name (derived from schema file name)

Namespace

One namespace object per XML Schema namespace is generated. Schema components that are not in any namespace are contained in a special namespace object with an empty NamespaceURI.
Note that for DTD, namespaces are also derived from attributes whose names begin with "xmlns".

Property	Type	Description
CodeName	string	Name for generated code (derived from prefix)
LocalName	string	Namespace prefix
NamespaceURI	string	Namespace URI
Types	Type collection	All types contained in this namespace
Library	Library	Library containing this namespace

Type

This object represents a complex or simple type. It is used to generate a class in the target language.
There is one additional type per library that represents the document, which has all possible root elements as members.
Anonymous types have an empty LocalName.

Property	Type	Description
CodeName	string	Name for generated code (derived from local name or parent declaration)
LocalName	string	Original name in the schema
Namespace	Namespace	Namespace containing this type
Attributes	Member collection	Attributes contained in this type*
Elements	Member colleciton	Child elements contained in this type
IsSimpleType	boolean	True for simple types, false for complex types
IsDerived	boolean	True if this type is derived from another type, which is also represented by a Type object
BaseType	Type	The base type of this type (if IsDerived is true)
IsDocumentRootType	boolean	True if this type represents the document itself
Library	Library	Library containing this type
IsFinal	boolean	True if declared as final in the schema
IsMixed	boolean	True if this type can have mixed content
IsAbstract	boolean	True if this type is declared as abstract
IsGlobal	boolean	True if this type is declared globally in the schema
IsAnonymous	boolean	True if this type is declared locally in an element

For simple types only:

Property	Type	Description
IsNativeBound	boolean	True if native type binding exists
NativeBinding	NativeBinding	Native binding for this type
Facets	Facets	Facets of this type
Whitespace	string	Shortcut to the Whitespace facet

* Complex types with text content (these are types with mixed content and complexType with simpleContent) have an additional unnamed attribute member that represents the text content.

Member

This object represents an attribute or element in the XML Schema. It is used to create class members of types.

Property	Type	Description
CodeName	string	Name for generated code (derived from local name or parent declaration)

© 2007 Altova GmbH

LocalName	string	Original name in the schema. Empty for the special member representing text content of complex types.
NamespaceURI	string	The namespace URI of this Element/Attribute within XML instance documents/streams.
DeclaringType	Type	Type originally declaring the member (equal to ContainingType for non-inherited members)
ContainingType	Type	Type where this is a member of
DataType	Type	Data type of this member's content
Library	Library	Library containing this member's DataType
IsAttribute	boolean	True for attributes, false for elements
IsOptional	boolean	True if minOccurs = 0 or optional attribute
IsRequired	boolean	True if minOccurs > 0 or required attribute
IsFixed	boolean	True for fixed attributes, value is in Default property
IsDefault	boolean	True for attributes with default value, value is in Default property
IsNillable	boolean	True for nillable elements
IsUseQualified	boolean	True if NamespaceURI is not empty
MinOccurs	integer	minOccurs, as in schema. 1 for required attributes
MaxOccurs	integer	maxOccurs, as in schema. 0 for prohibited attributes, -1 for unbounded
Default	string	Default value

NativeBinding

This object represents the binding of a simple type to a native type in the target programming language, as specified by the "schemanativetype" map.

Property	Type	Description
ValueType	string	Native type
ValueHandler	string	Formatter class instance

Facets

This object represents all facets of a simple type.
Inherited facets are merged with the explicitly declared facets. If a Length facet is in effect, MinLength and MaxLength are set to the same value.

Property	Type	Description
DeclaringType	Type	Type facets are declared on
Whitespace	string	"preserve", "collapse" or "replace"
MinLength	integer	Facet value
MaxLength	integer	Facet value
MinInclusive	integer	Facet value
MinExclusive	integer	Facet value
MaxInclusive	integer	Facet value
MaxExclusive	integer	Facet value

TotalDigits	integer	Facet value
FractionDigits	integer	Facet value
List	Facet collection	All facets as list

Facet

This object represents a single facet with its computed value effective for a specific type.

Property	Type	Description
LocalName	string	Facet name
NamespaceURI	string	Facet namespace
FacetType	string	one of "normalization", "lexicalspace", "valuespace-length", "valuespace-enum" or "valuespace-range"
DeclaringType	Type	Type this facet is declared on
FacetCheckerName	string	Name of facet checker (from schemafacet map)
FacetValue	string or integer	Actual value of this facet

Old object model (up to v2007)

The following objects are part of the old object model for code compatibility up to version 2007. They will be removed in a future release.

Namespace

Namespace abstraction. This object is part of the old object model for code compatibility up to version 2007.

Property	Type	Description
URI	string	The URI of this namespace
Prefix	string	The prefix of this namespace
ContainsPublicClasses	boolean	True if the Classes collection contains at least one non-internal Class object
Classes	Class collection	Collection of the classes in this namespace - step through them using **foreach**

Class

For every user-defined type declared in the schema (xsd) a class is generated. This object is part of the old object model for code compatibility up to version 2007.

Property	Type	Description
HasNamespace	boolean	True, if there is an associated Namespace object
Namespace	Namespace	The Namespace object this Class object is part of
NamespacePrefix	string	Prefix of the namespace in which the class is
NamespaceURI	string	URI of the namespace in which the class is
Name	string	Name of the type in the resulting file
SchemaName	string	Name of the type as in the original schema file

 © 2007 Altova GmbH

HasBaseObject	boolean	True if this type is derived from another type, which is also represented by a Class object.
BaseObject	Class	The base Class object if HasBaseObject is true
BaseNamespaceURI	string	Namespace URI of the base class
Base	string	Name of the base class
SchemaBase	string	Name of the base type as in the original schema file
BuiltInBase	string	Name of the root simple type
BuiltInSchemaBase	string	Name of the root simple type as in the original schema file
IsAbstract	boolean	True if this is an abstract type
IsSimpleType	boolean	True if this is a simple type
IsComplexFromSimpleType	boolean	True if this is a complex type and is derived from a simple type
IsComplexType	boolean	True if this is a complex type
IsSequence	boolean	True if the top-level group-type is "sequence"
IsChoice	boolean	True if the top-level group-type is "choice"
IsAll	boolean	True if the top-level group-type is "all"
Description	string	Description of this type. May contain line breaks.
IsGlobal	boolean	True if this type is declared globally in the schema
IsAnonymous	boolean	True if this type is declared locally in an element
IsInternal	boolean	True if this is the internal root pseudo-class that contains all global classes as members
Members	Member collection	A class representing a complexType contains one or more Members.
Facets	Facet collection	A class representing a simpleType or a complexType derived from a simpleType may contain Facets.

Member

Abstraction of a member-variable inside a class. Members are created for all children of a complex type. This object is part of the old object model for code compatibility up to version 2007.

Property	Type	Description
NamespaceURI	string	The namespace URI of this Element/Attribute within XML instance documents/streams.
Name	string	Name in the resulting file
SchemaName	string	Name as in the original schema file
XmlName	string	The name as it is expected to appear in XML instance documents/streams.
Type	string	The name of the class which represents the schema type
SchemaType	string	The schema type
TypeObject	Class	Generated class for this type. See explanation below *
HasTypeObject	boolean	True if TypeObject has a valid value
Description	string	Description of the Element/Attribute. Can contain line feeds.
IsBuiltinType	boolean	True if the type is a built-in schema type, e.g. string, unsignedInt, dateTime... It is the negation of HasTypeObject.

IsSimpleType	boolean	True if the type of this member is a simpleType
IsComplexFromSimpleType	boolean	True if the type of this member is a complex type and is derived from a simple type.
IsElement	boolean	True if this is an element
IsAttribute	boolean	True if this is an attribute
NodeType	string	"Element" or "Attribute"
MinOcc	integer	minOccurs, as in schema. 1 for required attributes
MaxOcc	integer	maxOccurs, as in schema. 0 for prohibited attributes
IsQualified	boolean	True if the form of the Element/Attribute is set to "qualified".
IsMixed	boolean	True if this element can have mixed content
HasTextValue	boolean	True if this member can contain text
Default	string	The default value defined in the schema
Fixed	string	The fixed value defined in the schema

*** TypeObject:**
For every type declared in the schema (xsd) a class is generated. All elements inside a complexType are generated as members of such classes. The types of these members can either be built-in types or user-defined types:

Built-in XML Schema types have no TypeObject, because they are mapped to predefined classes using the map type instruction.
For each member with a user-defined type, the TypeObject represents the class generated for this type.

Example:

```
<xs:complexType name="TypeA">
        <xs:sequence>
                <xs:element name="B" type="TypeB"/>
        </xs:sequence>
</xs:complexType>
<xs:complexType name="TypeB"/>
```

CodeGen would create two classes:

```
class TypeA
{
    ...
    TypeB  B;
}

class TypeB
{
    ...
}
```

Explanation:
When class "TypeA" is generated the member B is also generated. To find out the type of B, use the "TypeObject" method of the element which returns the class "TypeB". So we can get the name and use it there.

 © 2007 Altova GmbH

Facet

This class consists of the constraint itself, and several boolean variables. This object is part of the old object model for code compatibility up to version 2007.

The boolean variables tell you the kind of constraint you're currently dealing with. The names of the Boolean variables are identical to those used in the schema specification.

Property	Type	Description
Constraint	string	Holds the value of the facet
IsLength	boolean	
IsMinLength	boolean	
IsMaxLength	boolean	
IsMinInclusive	boolean	
IsMinExclusive	boolean	
IsMaxInclusive	boolean	
IsMaxExclusive	boolean	
IsTotalDigits	boolean	
IsFractionDigits	boolean	
IsWhiteSpace	boolean	
IsPattern	boolean	
IsEnumeration	boolean	
Enumeration	Enumeration collection	Holds a collection of Enumeration objects if this facet is of type enumeration.
Pattern	Pattern collection	Holds a collection of Enumeration objects if this facet is of type pattern.

Enumeration

Abstraction of an enumeration entry inside a facet. This object is part of the old object model for code compatibility up to version 2007.

Property	Type	Description
Index	integer	Holds the index of this enumeration value, starting with 0
Value	string	Holds an enumeration value

Pattern

Abstraction of a pattern entry inside a facet. This object is part of the old object model for code compatibility up to version 2007.

Property	Type	Description
Index	integer	Holds the index of this pattern value, starting with 0
Value	string	Holds a pattern value

14.8 Error Codes

Operating System Error Codes

201 File not found: '%s'
202 Cannot create file '%s'
203 Cannot open file '%s'
204 Cannot copy file '%s' to '%s'

Schema Error Codes

302 Validator: %s
303 Validator cannot load schema '%s'

Syntax Error Codes

401 Keyword expected
402 '%s' expected
403 No output file specified
404 Unexpected end of file
405 Keyword not allowed

Runtime Error Codes

501 Unknown variable '%s'
502 Redefinition of variable '%s'
503 Variable '%s' is not a container
504 Unknown property '%s'
505 Cannot convert from %s to %s
507 Unknown function
508 Function already defined
509 Invalid parameter
510 Division by zero
511 Unknown method
512 Incorrect number of parameters
513 Stack overflow

© 2007 Altova GmbH

15 Benutzerreferenz

Im Referenzteil der Dokumentation finden Sie eine Beschreibung aller XMLSpy Menübefehle und deren Verwendung. Wir haben versucht, das Handbuch so umfassend wie möglich zu gestalten. Sollten Sie trotzdem Fragen zu Themen haben, die in der vorliegenden Dokumentation nicht behandelt werden, werfen Sie bitte einen Blick in die FAQs und die Diskussionsforen auf der Altova Website. Sollten Ihre Fragen damit noch immer nicht beantwortet sein, kontaktieren Sie uns bitte über unser Support Center auf der Altova Website.

Bitte beachten Sie, dass im Menü Datei und Bearbeiten alle Windows-Standardbefehle und alle zusätzlichen Befehle im Zusammenhang mit XML- und Internetfunktionalitäten unterstützt werden.

15.1 Menü "Datei"

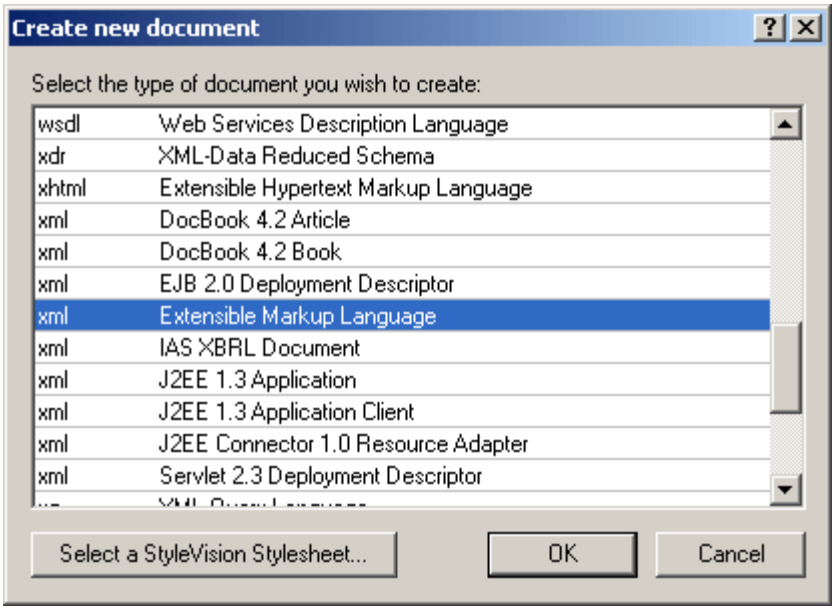
Das Menü "Datei" enthält alle Befehle, die zum Arbeiten mit Dateien benötigt werden. Die Befehle sind in der aus Windows Software-Produkten bekannten Reihenfolge angeordnet.

Zusätzlich zu den Standardbefehlen Neu, Öffnen, Speichern, Drucken, Druckereinrichtung, und Beenden bietet XMLSpy eine Reihe von XML- und applikationsspezifischen Befehlen.

15.1.1 Neu...

Strg+N

Der Befehl **Neu…** wird verwendet, um ein neues Dokument zu erstellen. Wenn Sie auf Neu… klicken, wird das Dialogfeld "Neues Dokument anlegen" geöfffnet, in dem Sie auswählen können, welche Art von Dokument erstellt werden soll. Falls das Dokument, das Sie erstellen wollen, nicht aufgelistet ist, wählen Sie "XML", und ändern Sie die Erweiterung des Dokuments beim Speichern. Beachten Sie, dass Sie neue Dateiarten in die Liste einfügen können, indem Sie diese unter Extras | Optionen auf dem Register "Dateiarten" eintragen.

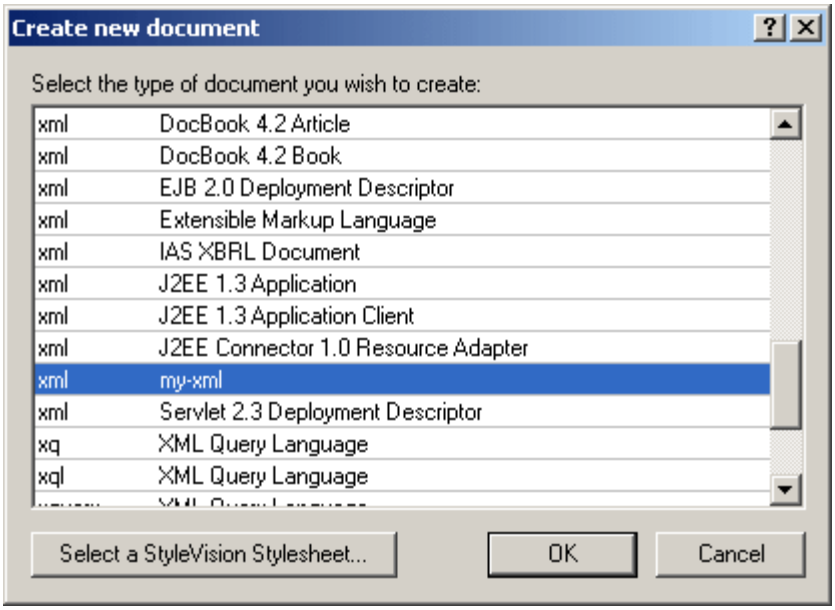


Erstellen von Vorlagen für neue Dokumente
Sie können mehrere Vorlagen für verschiedene Dateiarten erstellen, die Sie anschließend direkt über das Dialogfeld "Neues Dokument anlegen" öffnen und bearbeiten können. Um Ihre eigene Vorlage zu erstellen, sodass Sie in diesem Dialogfeld in der Liste der Dokumente angezeigt wird, müssen Sie zuerst ein Vorlagendokument erstellen und dieses dann in dem Ordner, der alle Vorlagen enthält, speichern. Gehen Sie vor wie folgt:

1. Öffnen Sie den Ordner **`XMLSpy\Template`** über den Windows Explorer oder Ihr bevorzugtes Navigationstool und wählen Sie aus den mit **`new. xxx`** benannten Dateien (wobei `. xxx` eine Dateierweiterung wie z.B. `. xml` und `. xslt` ist) eine rudimentäre Vorlagendatei aus.
2. Öffnen Sie die Datei in XMLSpy und ändern Sie die Datei Ihren Wünschen entsprechend. Diese Datei wird dann als Vorlage verwendet.
3. Wählen Sie anschließend **Datei | Speichern unter...** um die Datei wieder im Ordner \Template unter einem passenden Namen z.B. my-xml.xml zu speichern. Sie haben

 © 2007 Altova GmbH

nun eine Vorlage mit dem Namen `my-xml.xml`, die im Dialogfeld "Neues Dokument anlegen" in der Dateiliste angezeigt wird.

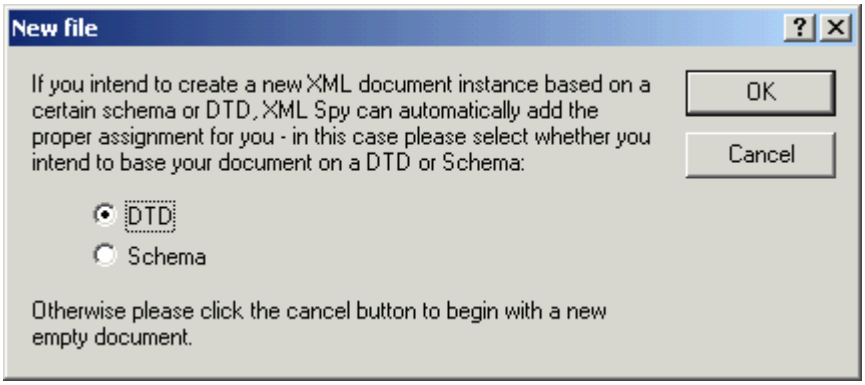


4. Zum Öffnen der Vorlage wählen Sie **Datei | Neu** und anschließend die Vorlage (in diesem Fall `my-xml`).

Bitte beachten Sie: Um eine Vorlage zu löschen, öffnen Sie das Vorlagen-Verzeichnis und löschen Sie dort die gewünschte Vorlage.

DTD / XML-Schema einem neuen XML-Dokument zuweisen
Wenn Sie ein neues Dokument einer bestimmten Art erstellen, das auf einem Standardschema (DTD oder XML-Schema) basiert, wird das Dokument automatisch mit der korrekten DTD oder XML-Schema-Verknüpfung geöffnet. So wird z.B. eine XHTML-Datei mit der damit verknüpften DTD `http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd` geöffnet. Eine XML-Schema-Datei (`.xsd`) wird mit dem Schemadokument `http://www.w3.org/2001/XMLSchema` verknüpft.

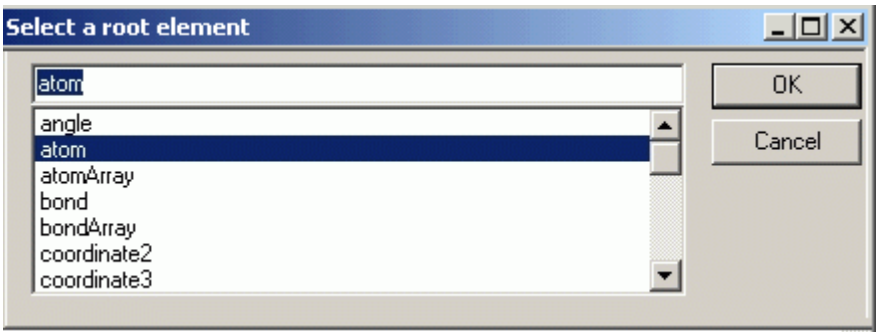
Wenn Sie eine neue Datei erstellen, für die das Schema nicht bekannt ist (z.B. eine XML-Datei), werden Sie aufgefordert, ein Schema (DTD oder XML-Schema) mit dem zu erstellenden Dokument zu verknüpfen.



Wenn Sie eine DTD oder ein XML-Schema mit Ihrem Dokument verknüpfen, klicken Sie im Dialogfeld "Neue Datei" auf OK, um den Pfad zum Schema anzugeben. Wenn Sie in diesem

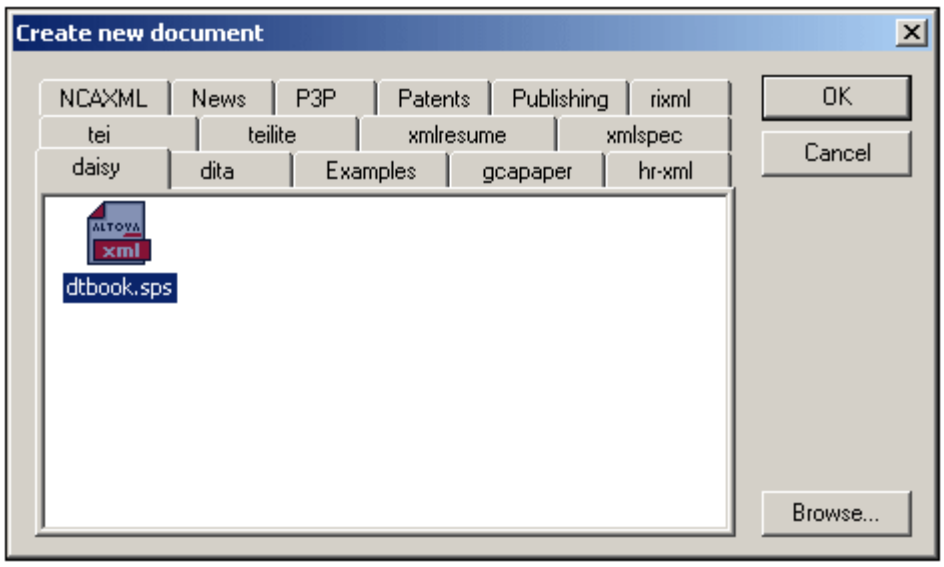
Dialogfeld auf "Abbrechen" klicken, wird eine neue Datei erstellt, die mit keinem Schema verknüpft ist.

Angabe des Dokumentelements eines neuen XML-Dokuments
Wenn Sie ein XML-Schema auswählen, kann dieses mehr als ein globales Element enthalten. Jedes davon kann potenziell ein Dokument- (oder Root) Element sein. Im Dialogfeld "Wählen Sie das Root-Element" können Sie das Root-Element für das XML-Dokument auswählen. Dieses Dialogfeld wird angezeigt, wenn Sie im Dialogfeld "Neue Datei" Schema auswählen und wenn das XML-Schema mehrere globale Elemente enthält.



Das neue XML-Dokument wird mit diesem Element als Dokument-Element erstellt.

Zuweisen eines StyleVision Power Stylesheet beim Erstellen eines neuen Dokuments
Bei Erstellung eines neuen XML-Dokuments können Sie es mit einem StyleVision Power Stylesheet (. `sps`-Datei) verknüpfen, um es in der Authentic-Ansicht anzeigen zu können. Wenn Sie im Dialogfeld "Neues Dokument anlegen" (*siehe Abbildung oben*) auf die Schaltfläche "StyleVision Stylesheet auswählen" klicken, erscheint das Dialogfeld "Neues Dokument anlegen" (*siehe Abbildung unten*).



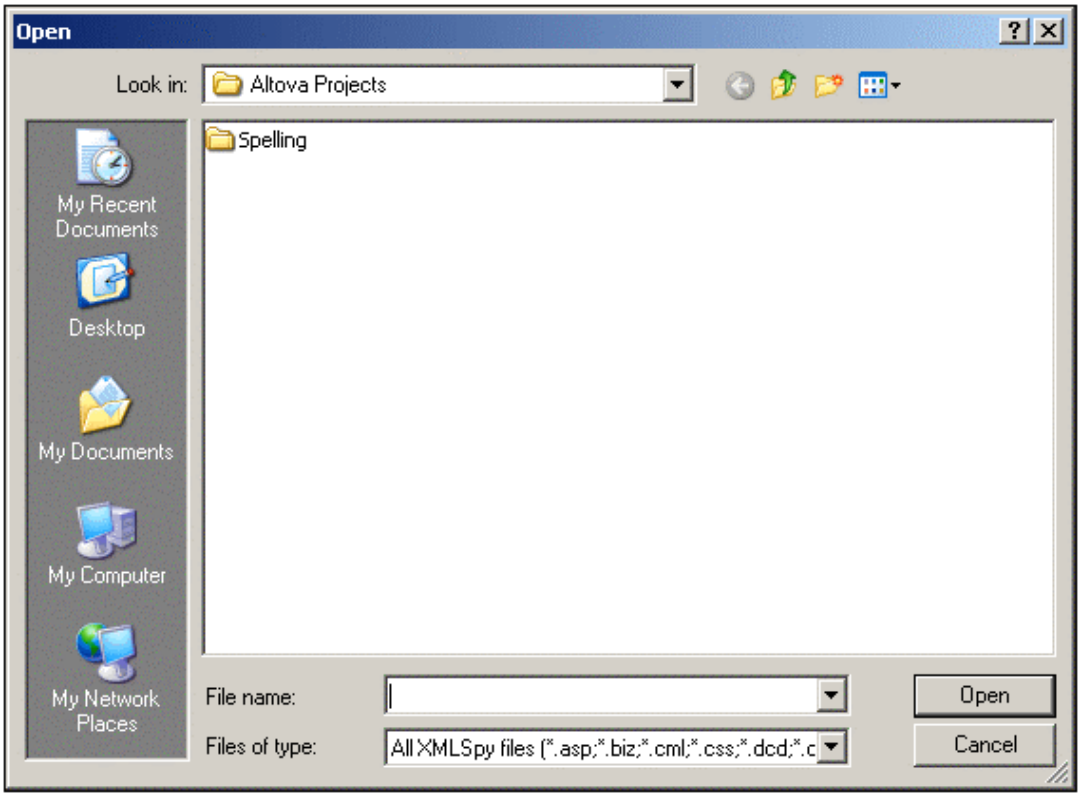
Auf den verschiedenen Ordner-Registern dieses Dialogfelds stehen Ihnen die gewünschten StyleVision Power Stylesheets zur Verfügung. Alternativ dazu können Sie das gewünschte StyleVision Power Stylesheet auch über die Schaltfläche **Durchsuchen…** suchen. Die Register in diesem Dialogfeld entsprechen den Ordnern im Ordner `sps/Template` im Programmordner der Applikation.

© 2007 Altova GmbH

15.1.2 Öffnen...

Strg+O

Mit "Öffnen..." rufen Sie das bekannte Windows-Dialogfeld "Öffnen" (*siehe Abbildung unten*) auf, über das Sie jedes XML-Dokument oder jede Textdatei öffnen können. Sie haben auch die Möglichkeit, mehrere Dateien auszuwählen. Im Listenfeld "Dateityp" können Sie auswählen, welche Dateien im Dialogfeld angezeigt werden sollen (die Liste der verfügbaren Dateiarten kann auf dem Register "Dateiarten" unter Extras | Optionen eingestellt werden).



Beim Öffnen einer XML-Datei wird diese auf Wohlgeformtheit überprüft. Ist sie nicht wohlgeformt, wird eine Fehlermeldung angezeigt. Beheben Sie den Fehler und klicken Sie auf Recheck, um die Datei erneut zu überprüfen. Wenn Sie die Option Validierung beim Öffnen der Datei aktiviert haben und die Datei ungültig ist, wird ebenfalls eine Fehlermeldung angezeigt. Beheben Sie den Fehler und klicken Sie auf Revalidate, um das Dokument erneut zu validieren.

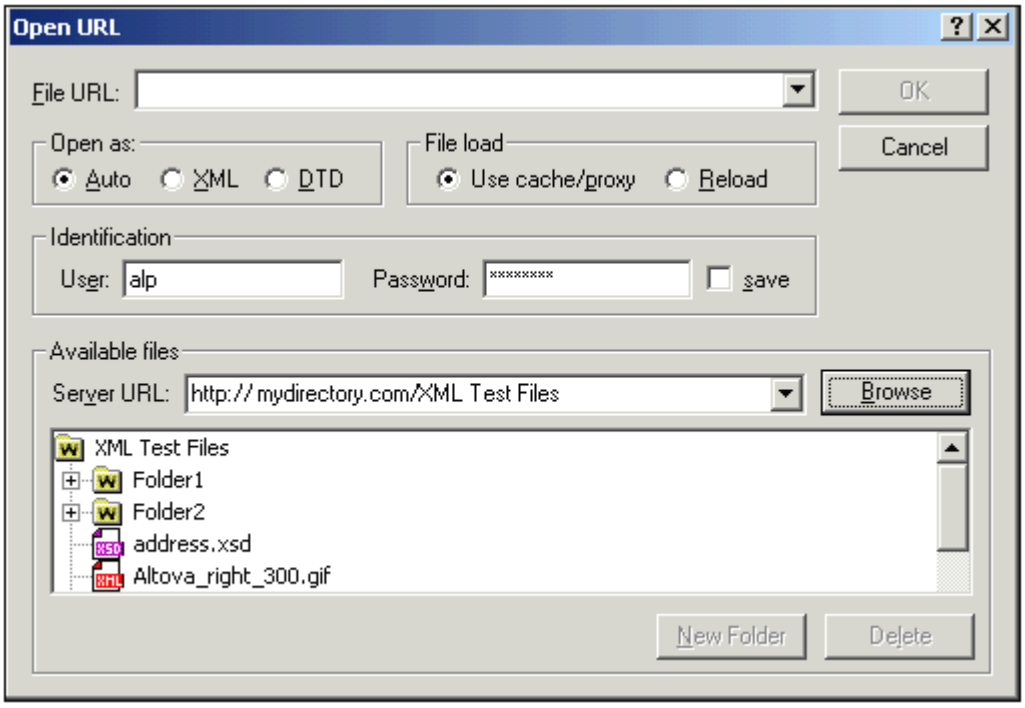
15.1.3 URL öffnen...

Mit **URL öffnen...** können Sie Dateien, die nicht lokal abgelegt sind, aus einer URL (Uniform

Resource Locator) mittels Http und WebDAV öffnen.

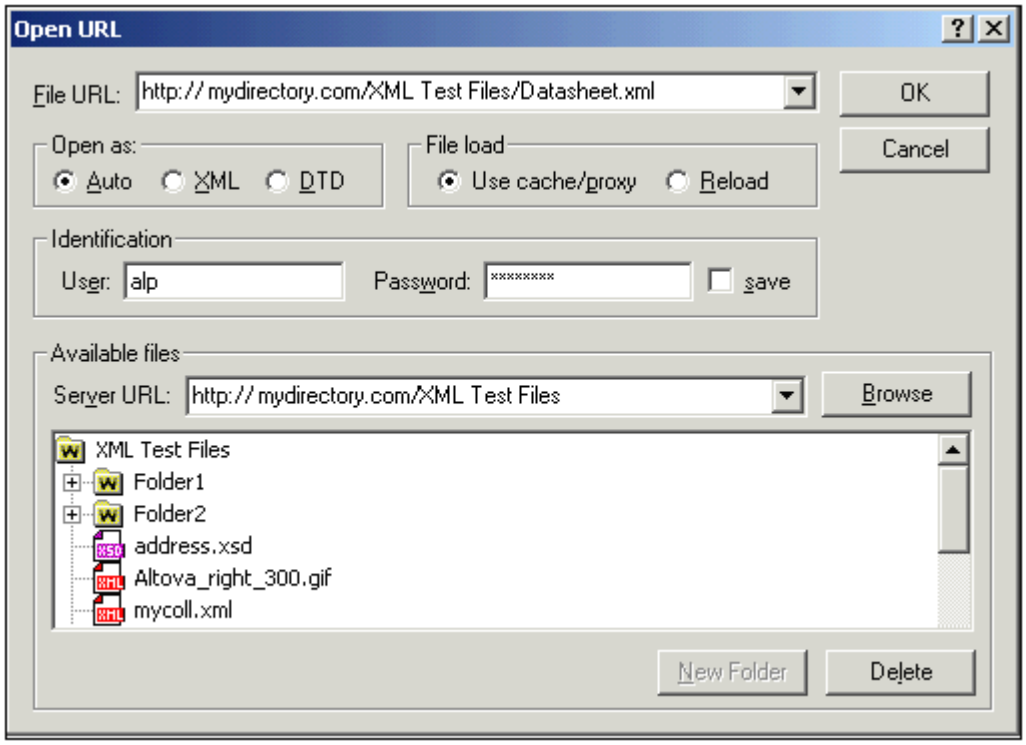
So öffnen Sie eine URL Adresse:

1. Klicken Sie auf den Befehl "URL öffnen". Daraufhin wird das Dialogfeld "URL öffnen" angezeigt.



2. Geben Sie die URL im Feld "Server URL" ein.
3. Falls der Server passwortgeschützt ist, geben Sie Ihren Benutzernamen und Ihr Passwort in den entsprechenden Feldern ein.
4. Klicken Sie auf die Schaltfläche **Durchsuchen** um durch die Verzeichnisstruktur des Servers zu navigieren.
5. Klicken Sie auf die in XMLSpy zu ladende Datei.

© 2007 Altova GmbH



Im Feld "Datei-URL" wird die URL der Datei angezeigt. Jetzt ist auch die Schaltfläche OK aktiv.

5. Klicken Sie auf **OK** um die Datei zu laden.
   Die Datei wird nun im Hauptfenster angezeigt.

Bitte beachten Sie: Die Funktion **Durchsuchen** ist nur auf Servern, die die Protokolle FTP, HTTP und HTTPS (bei Unterstützung von WebDAV) und auf Servern, die WebDAV unterstützen, verfügbar.

Datei laden
Um Ihnen mehr Kontrolle über den Ladeprozess zu geben, können Sie die Datei über den lokalen **Cache** oder den **Proxyserver** laden (was den Prozess beschleunigt, wenn die Datei zuvor schon einmal geladen worden ist). Alternativ dazu können Sie die Datei, wenn Sie z.B. mit einem Datenbanksystem oder einem Electronic Publishing-System arbeiten, über die Option "Neu laden" auch erneut laden.

Speicherschnittstellen
XMLSpy unterstützt drei Arten von Speicherschnittstellen:

- Generische Versionskontrollsysteme wie Microsoft Visual Source-Safe (oder kompatible Produkte)
- Generische Webserver wie WebDAV und FTP
- Spezielle XML- oder Schema-Speicherschnittstellen, wie Tamino, TEXTML Server, Virtuoso oder XML Canon.

XMLSpy unterstützt derzeit Source-Safe, FTP und WebDAV Server sowie Virtuoso über WebDAV. Zukünftige Versionen werden weitere Schnittstellen unterstützen.

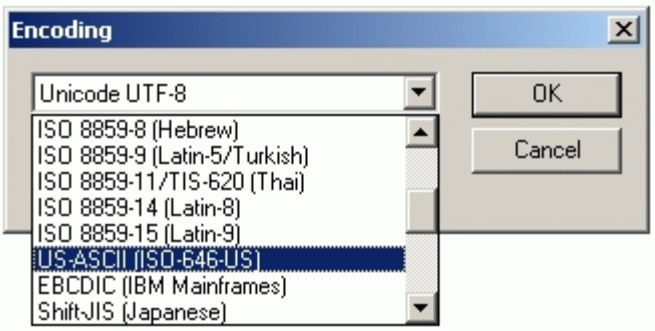
15.1.4 Neu laden

Mit **Neu laden** können Sie geöffnete Dokumente erneut laden.

Dieser Befehl ist nützlich, wenn ein geöffnetes Dokument außerhalb von XMLSpy geändert wurde. In diesem Fall werden Sie von XMLSpy gefragt, ob die Datei erneut geladen werden soll. Wenn Sie dies tun, gehen alle Änderungen, die Sie an der Datei seit der letzten Speicherung vorgenommen haben, verloren. Sie können diese Option im Dialogfeld "Optionen" (Extras | Optionen) ändern.

15.1.5 Kodierung

Mit **Kodierung...** können Sie die Kodierung der in Verwendung befindlichen Datei (XML oder nicht XML) ansehen und eine andere Kodierung auswählen, mit der das aktive Dokument beim nächsten Mal gespeichert wird.



Wenn Sie eine andere Kodierung als die derzeit verwendete auswählen, wird die Kodierungsspezifikation der XML-Deklaration entsprechend geändert. Bei 2-Byte und 4-Byte-Zeichenkodierungen (UTF-16, UCS-2 und UCS-4) können Sie weiters die zu verwendende Bytefolge in der Datei angeben. Eine andere Methode, um die Kodierung eines XML-Dokuments zu ändern, ist, das Kodierungsattribut direkt in der XML-Deklaration des Dokuments zu ändern.

Die Standardkodierung für vorhandene und neue XML- und nicht-XML-Dokumente kann im Dialogfeld "Optionen" auf dem Register "Kodierung" festgelegt werden.

Anmerkung: Beim Speichern eines Dokumentes prüft XMLSpy automatisch die Kodierungsspezifikation und öffnet ein Dialogfeld, falls es den vom Benutzer angegebenen Kodierungsnamen nicht erkennt. Sollte Ihr Dokument Zeichen enthalten, die in der gewählten Kodierung nicht angezeigt werden können, erhalten Sie eine Warnmeldung beim Speichern der Datei.

15.1.6 Schließen

Mit **Schließen** beenden Sie das aktive Dokument. Sollten Sie die Datei verändert haben (in der Titelleiste wird der Dateiname mit einem Sternchen "*" angezeigt), werden Sie gefragt, ob Sie die Datei vor dem Schließen speichern möchten.

15.1.7 Alle schließen

Mit **Alle Schließen** beenden Sie alle Dokumente. Sollten Sie Dateien verändert haben (in der Titelleiste wird der Dateiname mit einem Sternchen "*" angezeigt), werden Sie gefragt, ob Sie die Dateien vor dem Schließen speichern möchten.

© 2007 Altova GmbH

15.1.8 Speichern

Strg+S

Mit **Speichern** wird der Inhalt des aktiven Dokuments in der geöffneten Datei gespeichert. Beim Speichern wird das Dokument automatisch auf Wohlgeformtheit geprüft. Außerdem wird das Dokument auch automatisch validiert, wenn diese Option im Dialogfeld "Optionen" (**Extras | Optionen**) auf dem Register "Datei" aktiviert wurde. Die XML Deklaration wird auch auf Kodierungsspezifikationen geprüft. Diese Kodierung wird beim Speichern des Dokuments verwendet.

15.1.9 Speichern unter...

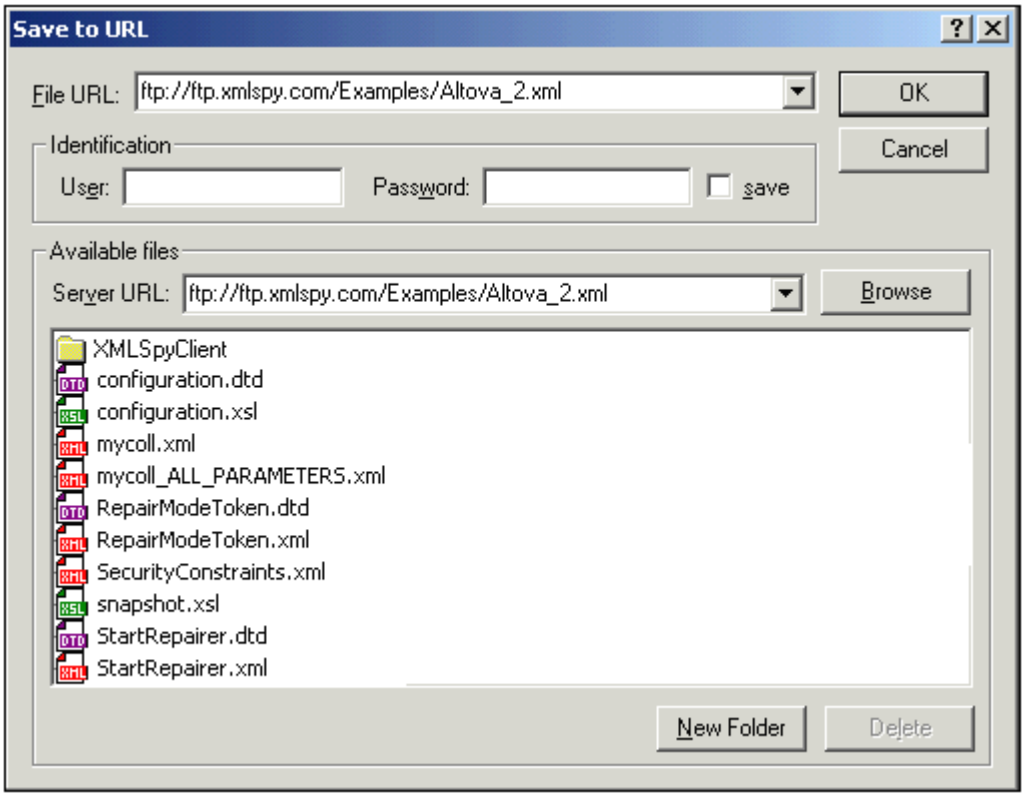
Mit dem Befehl **Speichern unter** gelangen Sie zu dem aus Windows bekannten Dialogfeld "Speichern unter...". Hier geben Sie den Namen und den Ordner an, in dem die Datei gespeichert werden soll. Bei Verwendung des Befehls **Speichern unter** werden dieselben Validierungen und Prüfungen durchgeführt, wie bei dem Befehl **Speichern.**

15.1.10 Speichern unter URL...

Mit "Speichern unter URL..." können Sie Dateien, die nicht lokal abgelegt sind, unter einer URL- (Uniform Resource Locator) Adresse speichern.

So speichern Sie ein XML-Dokument unter einer URL-Adresse:

1. Die zu speichernde Datei muss im Hauptfenster aktiv sein.
2. Wählen Sie die Menüoption **Datei | Speichern unter URL.**



3. Klicken Sie auf die Schaltfläche **Durchsuchen** um durch die Verzeichnisstruktur auf dem Server zu navigieren.
4. Geben Sie den Namen im Feld Datei-URL ein, oder markieren Sie einen Eintrag im Listenfeld "Vorhandene Dateien", wenn Sie die Datei überschreiben möchten.

Bitte beachten Sie:

- Die Funktion **Durchsuchen** ist nur für Server verfügbar, die folgende Protokolle unterstützen: FTP, HTTP und HTTPS (falls der Server WebDAV unterstützt) und auf Servern, die WebDAV unterstützen.
- Falls der Server, mit dem Sie sich verbinden möchten, passwortgeschützt ist, geben Sie bitte die benötigten Daten ein, und klicken Sie zum Verbinden auf die Schaltfläche **OK**. Sie können Ihre Daten auch in den Feldern "Benutzer" und "Passwort" eingeben. Mit dem Kontrollkästchen "Speichern" werden die Passwortdaten für spätere Logins gespeichert.

Speicherschnittstellen
XMLSpy unterstützt drei Arten von Speicherschnittstellen:

- Generische Versionskontrollsysteme wie Microsoft Visual Source-Safe (oder kompatible Produkte)
- Generische Webserver wie WebDAV und FTP
- Spezielle XML- oder Schema-Speicherschnittstellen, wie Tamino, TEXTML Server, Virtuoso oder XML Canon.

XMLSpy unterstützt derzeit Source-Safe, FTP und WebDAV Server, sowie Virtuoso über WebDAV. Zukünftige Versionen werden weitere Schnittstellen unterstützen.

15.1.11 Alles speichern

© 2007 Altova GmbH

Mit diesem Befehl werden alle Änderungen gespeichert, die Sie an offenen Dokumenten durchgeführt haben. Dieser Befehl ist hilfreich, wenn Sie gleichzeitig an mehreren Dokumenten arbeiten. Falls ein zu speicherndes Dokument bisher noch nie gespeichert wurde, wird das Dialogfeld Speichern unter... (z.B. ein über den Befehl Neu erstelltes Dokument) für dieses Dokument angezeigt.

15.1.12 Als Mail senden...

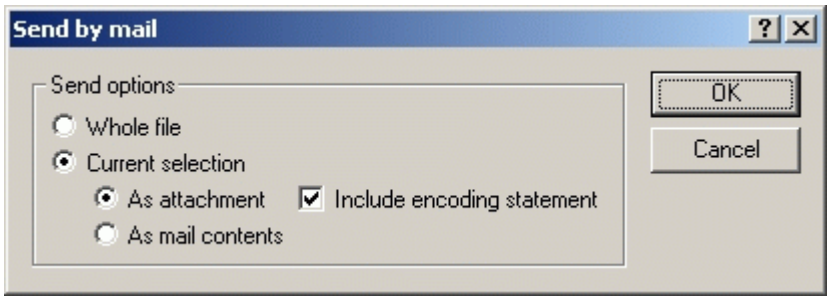
Mit dem Befehl **Als Mail senden...** können Sie jede XML-Datei oder Teile einer XML-Datei als Email versenden. Sie haben folgende Möglichkeiten:

- Versenden eines XML-Dokuments als Anhang oder als Inhalt einer E-Mail.
- Versenden eines Teils eines XML-Dokuments als Anhang oder Inhalt einer E-Mail.
- Versenden einer Gruppe von Dateien (die im Projektfenster ausgewählt wurden) als Anhang einer E-Mail.
- Versenden einer URL (die im Projektfenster ausgewählt wurde) als Anhang oder als Link.

Bitte beachten Sie: Zur Verwendung dieser Funktion benötigen Sie ein MAPI-kompatibles E-Mail-System.

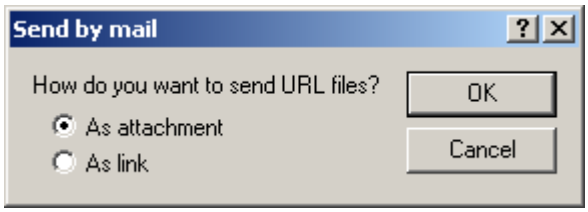
Versenden von Dokumenten und Teilen eines Dokuments
Um ein XML-Dokument zu versenden,

1. Machen Sie es im Haupfenster zum aktiven Dokument. Wenn Sie nur einen Teil aus einem XML-Dokument oder eine Gruppe von Dateien eines Projekts versenden möchten, markieren Sie den Abschnitt im Dokument oder wählen Sie die gewünschten Dateien im Projekt aus.
2. Klicken Sie nun auf **Als Mail senden...**. Daraufhin wird das folgende Dialogfenster angezeigt.



3. Aktivieren Sie die gewünschten Optionen und klicken Sie auf **OK**. Das/die ausgewählte(n) Dokument(e)/der ausgewählte Inhalt/die ausgewählten URLs werden an die E-Mail angehängt bzw. der Inhalt wird in die E-Mail eingefügt.

URL per Mail versenden
Zum Versenden einer oder mehrerer URLs wählen Sie die URLs im Projektfenster aus und klicken Sie auf **Als Mail senden...**. Es wird das folgende Dialogfenster angezeigt.



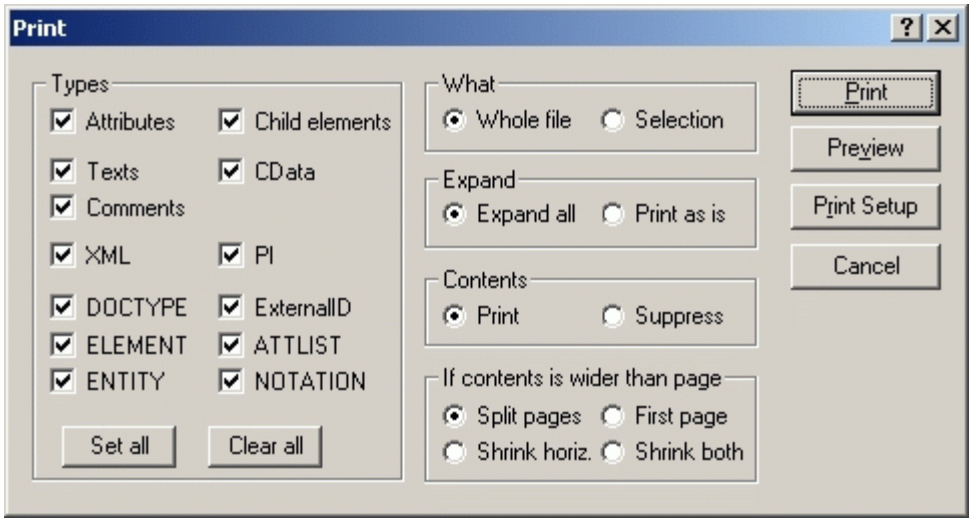
Wählen Sie aus, in welcher Form die URL versendet werden soll und bestätigen Sie mit **OK**.

15.1.13 Drucken...

Strg+P

Der Befehl **Drucken...** öffnet das Dialogfeld "Drucken". Hier können Sie festlegen, was die Druckausgabe enthalten soll.

Wenn Sie in der Enhanced Grid-Ansicht auf **Drucken...** klicken, wird ein Dialogfeld mit Druckoptionen geöffnet (*siehe Abbildung unten*), in dem Sie Druckoptionen für das XML-Dokument einstellen können. Durch Klicken auf die Schaltfläche "Drucken" wird das Dialogfeld für die Druckeroptionen geöffnet.



Im Folgenden sind die Optionen für das Drucken in der Enhanced Grid-Ansicht beschrieben:

- Im Bereich **Typ** haben Sie die Möglichkeit festzulegen, was in der Druckausgabe enthalten sein soll.
- Unter **Drucken** können Sie auswählen, ob die gesamte Datei oder ein ausgewählter Bereich gedruckt werden soll.
- Im Bereich **Erweitern** geben Sie an, ob das Dokument so gedruckt werden soll, wie es ist, oder ob Sie alle Child-Elemente ausgeklappt sehen möchten.
- Im Bereich **Inhalt** können Sie entscheiden, ob Sie den Inhalt aller Nodes oder nur die Node-Namen drucken möchten.
- Im Bereich **Wenn Inhalt breiter als Seite** können Sie entscheiden, was geschehen soll, wenn das Dokument größer als eine Papierseite ist. Die Option *aufteilen* druckt das gesamte Dokument in normaler Größe, und der Inhalt wird auf mehrere Seiten aufgeteilt (sowohl senkrecht als auch waagerecht). Sie können die Seiten dann zu einem Poster zusammenfügen. Die Option *Erste Seite* ermöglicht es Ihnen, nur die erste linke Seite des Druckbereichs zu drucken. Der restliche Bereich, der über den

 © 2007 Altova GmbH

Seitenrand ragt, wird nicht gedruckt. Diese Option eignet sich v.a. dann, wenn sich die meisten wichtigen Informationen in der Grid-Ansicht des Dokuments auf der linken Seite befinden. Die Option *Horizontal verkleinern* verkleinert die Druckausgabe bis sie horizontal auf der Papierseite Platz hat. Die Druckausgabe kann mehrere Seiten lang sein. Die Option *Beide verkleinern* reduziert die Größe der Druckausgabe bis sie ganz auf einer Papierseite Platz hat.

- Mit der Schaltfläche **Drucken** wird das Dokument nach Ihren Vorgaben ausgedruckt.
- Mit der Schaltfläche **Vorschau** erhalten Sie eine Vorschau auf die Druckausgabe, bevor diese auf Papier gedruckt wird.
- Mit der Schaltfläche **Drucker einrichten** wird das Dialogfeld "Druck einrichten" geöffnet, wo Sie Papierformat, Ausrichtung, und weitere Druckeinstellungen für diesen Ausdruck vornehmen können. Siehe auch Befehl "Drucker einrichten".

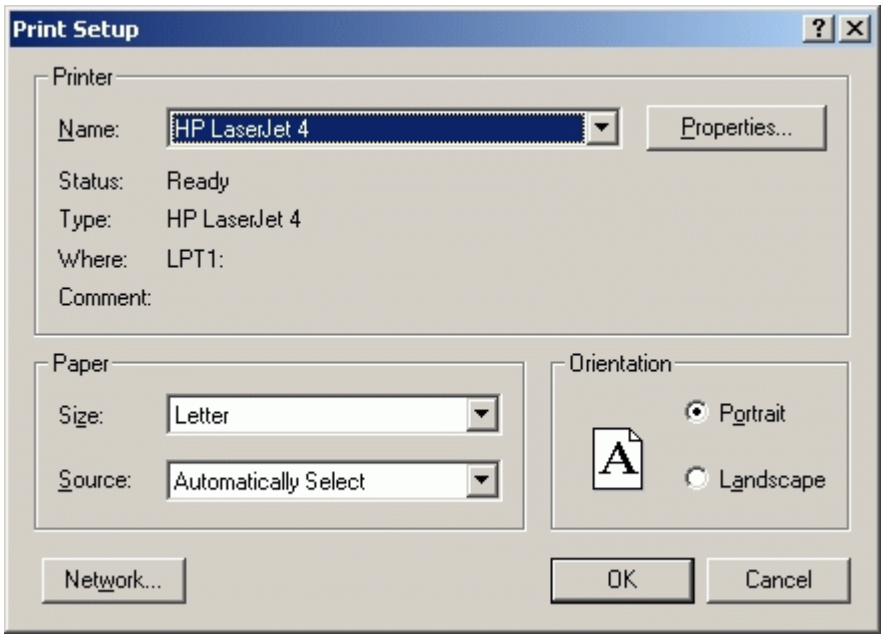
Bitte beachten Sie: Sie können die Spaltenbreite in der Grid-Ansicht anpassen, um die Druckausgabe zu optimieren.

15.1.14 Druckvorschau

Mit dem Befehl **Druckansicht** öffnen Sie das Dialogfeld "Drucken". Klicken Sie auf die Schaltfläche "Vorschau", um eine Druckvorschau des aktiven Dokuments zu sehen.

15.1.15 Druckereinrichtung...

Mit dem Befehl **Druckereinrichtung...** gelangen Sie in das druckerspezifische Dialogfeld, wo Sie Papierformat, Ausrichtung und weitere Einstellungen tätigen können, die für alle zukünftigen Druckaufträge gültig sind.



Das obige Bild zeigt das Dialogfeld, bei dem ein Drucker des Typs HP Laserjet 4 am parallelen Port LPT1 angeschlossen ist.

15.1.16 Zuletzt verwendete Dateien

Im Menü **Datei** sehen Sie eine Liste der neun zuletzt verwendeten Dateien, wobei die zuletzt verwendete Datei zuoberst angezegit wird. Durch Anklicken einer Datei können Sie diese öffnen. Über die Tastatur können Sie mit der Funktionstaste **ALT+A** das Menü **Datei** aufrufen, und dann die der Datei entsprechende Nummer eintippen.

15.1.17 Beenden

Mit diesem Befehl beenden Sie XMLSpy, nachdem Sie Ihre Arbeit an Ihren Dokumenten abgeschlossen haben. Wenn noch nicht gespeicherte Dokumente offen sind, werden Sie gefragt, ob Sie die Änderungen speichern möchten.
XMLSpy speichert weiters Programmeinstellungen, sowie Informationen über die zuletzt verwendeten Dateien.

© 2007 Altova GmbH

15.2 Menü "Bearbeiten"

In diesem Menü befinden sich Befehle, die beim Editieren mit XMLSpy verwendet werden.

Zusätzlich zu den Standardbefehlen Rückgängig, Wiederherstellen, Ausschneiden, Kopieren, Einfügen, Löschen, Alles markieren, Suchen, Weitersuchen und Ersetzen, bietet XMLSpy Sonderbefehle für folgende Funktionen:

- Auswahl in Zwischenablage als XML-Text kopieren,
- Als strukturierten Text kopieren, oder
- XPath Selector für das ausgewählte Objekt in die Zwischenablage kopieren.
- Einfügen und Entfernen von Lesezeichen und Gehe zu Lesezeichen

15.2.1 Rückgängig

Kürzel: **Strg + Z & Alt + Rück**

Dieser Befehl unterstützt unbegrenztes Rückgängigmachen aller Aktionen! Jede durchgeführte Aktion kann in Einzelschritten rückgängig gemacht werden. Der Verlauf der durchgeführten Aktionen wird beim Schließen des Dokuments gespeichert, sodass Aktionen auch zu einem späteren Zeitpunkt zurückverfolgt werden können.

15.2.2 Wiederherstellen

Kürzel: **Strg + Y**

Mit **Wiederherstellen** können Sie zuvor rückgängig gemachte Aktionen wiederherstellen. Sie können den Verlauf des Dokuments sowohl zurück als auch nach vorne verfolgen.

15.2.3 Ausschneiden

Kürzel: **Umschalt + Entf**, oder **Strg + X**

Mit diesem Befehl wird der makierte Text bzw. das markierte Objekt in die Zwischenablage kopiert und aus der aktuellen Position im Dokument gelöscht.

15.2.4 Kopieren

Kürzel: **Strg + C**

Mit diesem Befehl kopieren Sie markierten Text oder Objekte in die Zwischenablage. So können Sie Daten in XMLSpy duplizieren, oder in eine andere Applikation kopieren.

Bitte beachten Sie: Es gibt zwei verschiedene Befehle um Elemente in Textform in die Zwischenablage zu kopieren: Als XML-Text kopieren und Als strukturierten Text kopieren. Auf dem Register "Bearbeiten" unter Extras | Optionen können Sie festlegen, welche dieser beiden Operationen bei Auswahl des Befehls "Kopieren" durchgeführt werden soll.

15.2.5 Einfügen

Kürzel: **Strg + V**

Mit diesem Befehl fügen Sie den Inhalt der Zwischenablage an der aktuellen Cursorposition ein.

15.2.6 Löschen

Tastatur: **Entf**

Mit diesem Befehl wird der markierte Text/das markierte Objekt gelöscht, ohne dass es in die Zwischenablage kopiert wird.

15.2.7 Als XML-Text kopieren

Mit dem Befehl **Als XML-Text kopieren** können Sie auf einfache Weise Daten aus anderen Produkten, die Datenmanipulation auf XML-Ebene erlauben, kopieren. Beim Editieren Ihres Dokuments in der Grid-Ansicht können Sie Elemente in ihrer XML-Textdarstellung in die Zwischenablage kopieren:

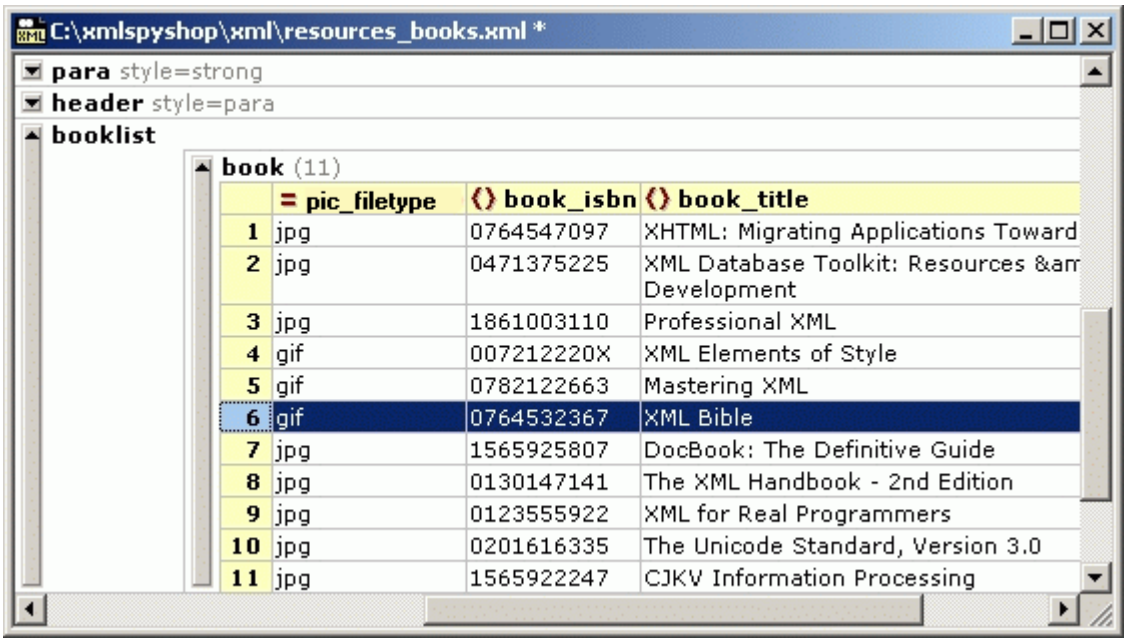
```
<row>
    <para align="left">
        <bold>Check the FAQ</bold>
    </para>
    <para>
        <link mode="internal">
            <link_section>support</link_section>
            <link_subsection>faq30</link_subsection>
            <link_text>XMLSPY 4.0 FAQ</link_text>
        </link>
        <link mode="internal">
            <link_section>support</link_section>
            <link_subsection>faq25</link_subsection>
            <link_text>XMLSPY 3.5 FAQ</link_text>
        </link>
    </para>
</row>
```

Der Befehl formatiert den Text automatisch nach den gültigen Einstellungen für das Speichern der Datei. Diese Einstellungen können im Dialogfeld "Optionen" (**Extras | Optionen**) auf dem Register "Datei" im Bereich "Datei speichern" vorgenommen werden.

15.2.8 Als strukturierten Text kopieren

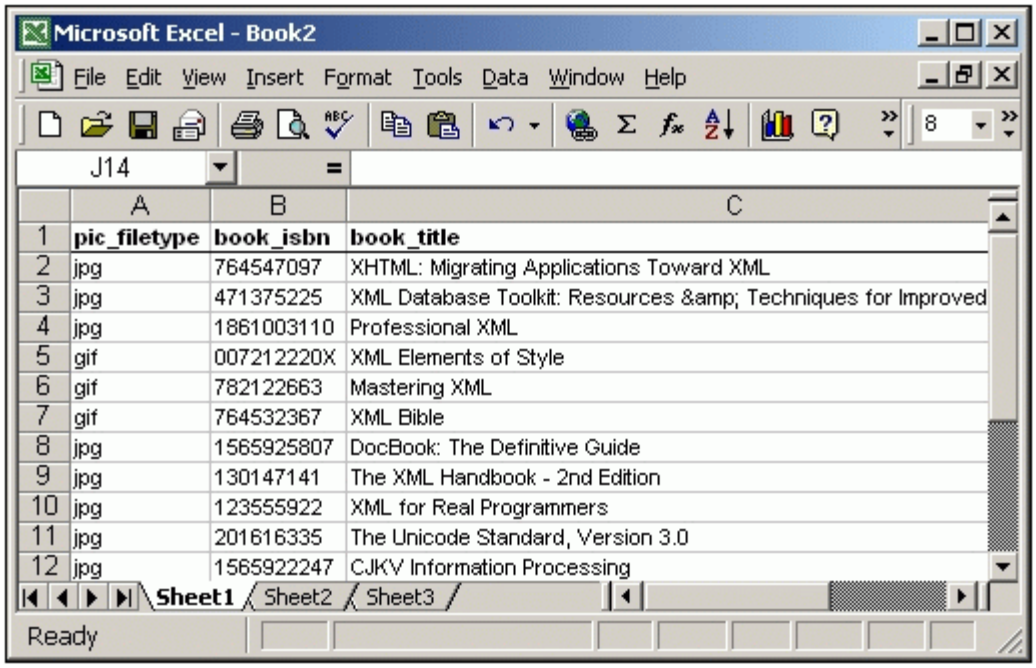
Mit diesem Befehl werden Elemente so in die Zwischenablage kopiert, wie sie am Bildschirm angezeigt werden. Dieser Befehl ist beim Kopieren von tabellenförmigen Daten in der Grid-Ansicht sehr nützlich. Die kopierten Daten können in XMLSpy sowie in Produkten von Drittanbietern verwendet werden. So können Sie z.B. XML-Daten in Spreadsheet-Applikationen wie Microsoft Excel kopieren.

© 2007 Altova GmbH



	pic_filetype	book_isbn	book_title
1	jpg	0764547097	XHTML: Migrating Applications Toward
2	jpg	0471375225	XML Database Toolkit: Resources &am Development
3	jpg	1861003110	Professional XML
4	gif	007212220X	XML Elements of Style
5	gif	0782122663	Mastering XML
6	gif	0764532367	XML Bible
7	jpg	1565925807	DocBook: The Definitive Guide
8	jpg	0130147141	The XML Handbook - 2nd Edition
9	jpg	0123555922	XML for Real Programmers
10	jpg	0201616335	The Unicode Standard, Version 3.0
11	jpg	1565922247	CJKV Information Processing

Wenn Sie diese Tabelle nach Excel kopieren, werden Ihre Daten wie folgt angezeigt:



	A	B	C
1	pic_filetype	book_isbn	book_title
2	jpg	764547097	XHTML: Migrating Applications Toward XML
3	jpg	471375225	XML Database Toolkit: Resources & Techniques for Improved
4	jpg	1861003110	Professional XML
5	gif	007212220X	XML Elements of Style
6	gif	782122663	Mastering XML
7	gif	764532367	XML Bible
8	jpg	1565925807	DocBook: The Definitive Guide
9	jpg	130147141	The XML Handbook - 2nd Edition
10	jpg	123555922	XML for Real Programmers
11	jpg	201616335	The Unicode Standard, Version 3.0
12	jpg	1565922247	CJKV Information Processing

Bitte beachten Sie: Das Ergebnis dieses Befehls hängt davon ab, wie die Daten am Bildschirm angezeigt werden. Wenn das sich das XML-Fragment, das als Beispiel für den Befehl Als XML-Text kopieren verwendet wurde, in der Tabellen-Ansicht der Grid-Ansicht befindet, sieht das Resultat des Kopiervorgangs aus wie folgt:

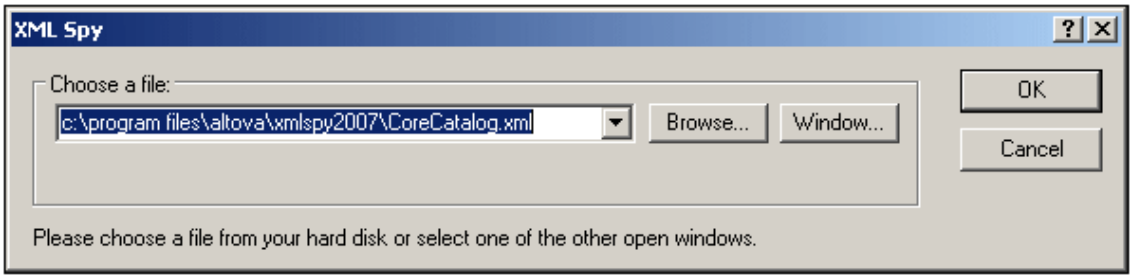
```
row
   para
      align    bold    link
      left     Check the FAQ
         link
            mode       link_section    link_subsection   link_text
            internal   support         faq30             XMLSPY 3.5 FAQ
            internal   support         faq25             XMLSPY 2.5 FAQ
```

In der normalen Grid-Ansicht werden die Daten wie folgt in die Zwischenablage kopiert:

```
row
    para
        align    left
        bold     Check the FAQ
    para
        link
            mode              internal
            link_section      support
            link_subsection   faq30
            link_text         XMLSPY 3.5 FAQ
        link
            mode              internal
            link_section      support
            link_subsection   faq25
            link_text         XMLSPY 2.5 FAQ
```

15.2.9 Dateipfad einfügen

Der Befehl **Dateipfad einfügen** ist in der Text- und der Grid-Ansicht von Dokumenten jedes Dateityps aktiv. Sie können damit den Pfad zu einer Datei am Cursorpunkt einfügen. Wenn Sie auf den Befehl klicken, erscheint ein Dialogfeld (*Abbildung unten*), in dem Sie die gewünschte Datei auswählen können.



Die gewünschte Datei kann auf zwei Arten ausgewählt werden: (i) durch Suchen nach der Datei (mit Hilfe der Schaltfläche **Durchsuchen***)*; (ii) durch Auswählen des Fensters, in dem die Datei geöffnet ist (Schaltfläche **Fenster**). Klicken Sie anschließend auf OK. Der Pfad zur ausgewählten Datei wird im aktiven Dokument an der Cursorposition eingefügt.

© 2007 Altova GmbH

15.2.10 XPath kopieren

Der Befehl **XPath kopieren** erstellt einen XPath-Ausdruck, der den/die aktuell ausgewählten Node(s) auswählt und den Ausdruck in die Zwischenablage kopiert, sodass Sie ihn dann in ein Dokument (z.B. ein XSLT-Dokument) einfügen können. Alle Ausdrücke beginnen beim Dokument-Root, z.B.:

- `/company/person/last` wenn der Element-Node `last` ausgewählt ist
- `/company/person` wenn ein oder mehrere Element-Nodes `person` ausgewählt sind

Der Befehl "XPath kopieren" ist nur in der Enhanced Grid-Ansicht aktiv.

15.2.11 Pretty-Print

Mit diesem Befehl wird Ihr XML-Dokument in der Text-Ansicht neu formatiert, sodass Sie eine strukturierte Ansicht Ihres Dokuments erhalten. Jeder Child Node wird um den Betrag eingerückt, den Sie im Dialogfeld "Optionen" (**Extras | Optionen**) auf dem Register "Datei" unter "Datei speichern" angegeben haben. Diese Funktion kann nur auf wohlgeformte Dokumente angewendet werden.

15.2.12 Alles markieren

Strg + A

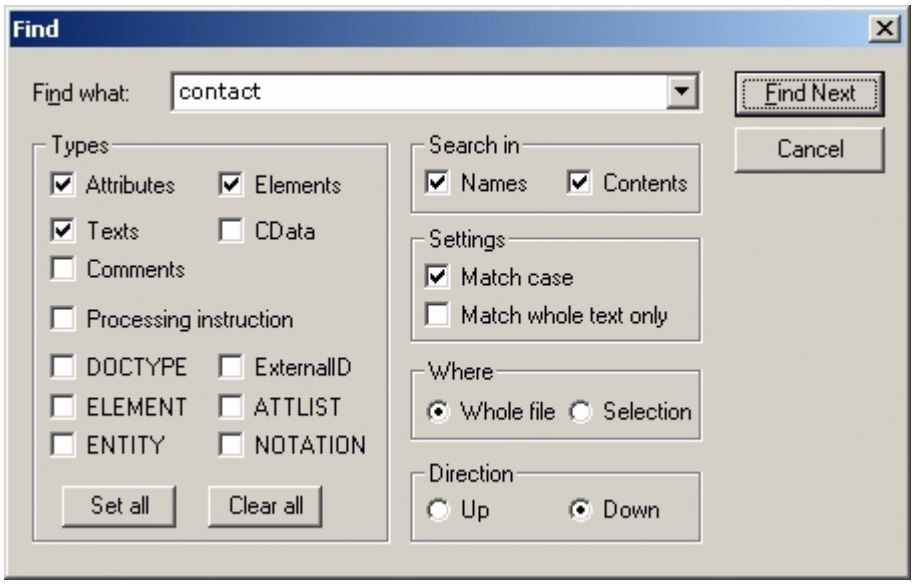
Mit diesem Befehl wird der Inhalt des gesamten Dokuments markiert.

15.2.13 Suchen...

Strg + F

Mit diesem Befehl rufen Sie das Dialogfeld "Suchen" auf, in dem Sie den zu suchenden String sowie weitere Suchoptionen spezifizieren können.
Abhängig von der ausgewählten Ansicht, gibt es im Dialogfeld "Suchen" verschiedene zur Auswahl stehende Optionen.
Um nach Text zu suchen, geben Sie diesen in das Listenfeld "Suchen nach" ein oder wählen Sie einen der letzten 10 Einträge aus der Dropdown-Liste aus und legen Sie anschließend die Suchoptionen fest.

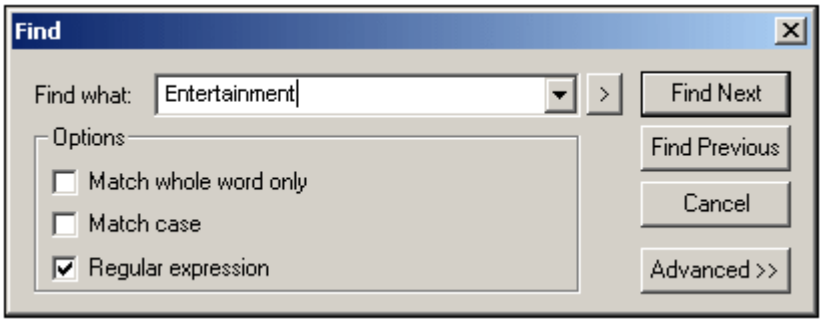
Grid-Ansicht
In der Grid-Ansicht erscheint das folgende Dialogfeld. Die verfügbaren Optionen sind unten beschrieben.



Aktivieren Sie die gewünschten Optionen oder klicken Sie auf ein Optionsfeld.

- Im Bereich "Typ" wird definiert, welche XML-Dokument-Nodes oder -Komponenten in die Suche einbezogen werden sollen. Hierdurch können Sie bestimmte Node-Typen bei der Suche überspringen. Durch Klicken auf die Schaltfläche "Alle aktiv" werden alle Kontrollkästchen im Bereich Typ aktiviert; die Schaltfläche "Zurücksetzen" deaktiviert alle Kontrollkästchen.
- Im Bereich "Suche in Element" können Sie definieren, ob nach Text nur in Node-Namen, dem Inhalt des Node oder in beidem gesucht werden soll.
- Im Bereich "Einstellungen" können Sie angeben, ob die Groß- bzw. Kleinschreibung beachtet werden soll, und ob nur nach ganzen Wörtern gesucht werden soll.
- Im Bereich "Suche in" wird definiert, in welchem Bereich die Suche erfolgen soll.
- Im Bereich "Richtung" definieren Sie, in welche Richtung die Suche erfolgen soll.

Text-Ansicht
In der Textansicht wird das folgende Dialogfeld angezeigt. Die verfügbaren Optionen sind im Folgenden beschrieben.



Die folgenden Suchoptionen stehen zur Verfügung:

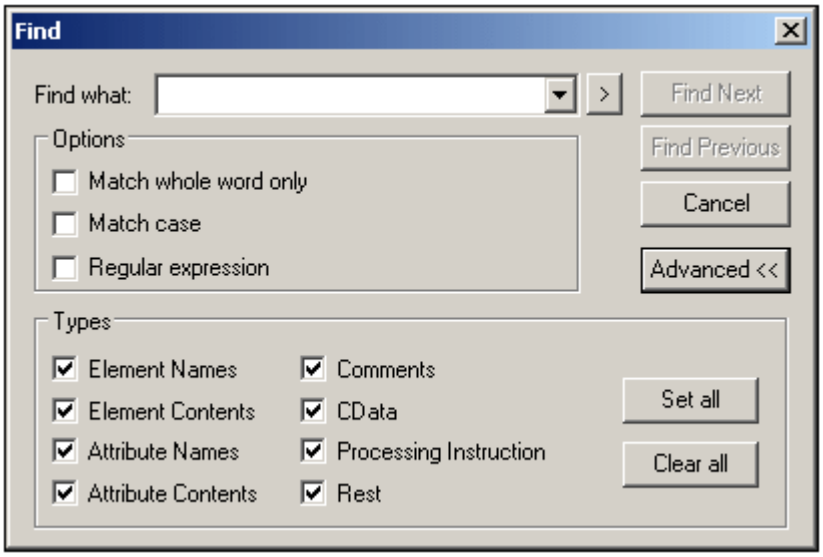
- Ganzes Wort: Bei der Suche werden nur die Wörter im Text gefunden, die exakt mit dem gesuchten Text übereinstimmen. Bei Eingabe des Suchstrings ﬁ würde z.B. bei Aktivierung der Option "Ganzes Wort" nur das Wort ﬁ gefunden; ﬁ in ﬂess würde nicht gefunden werden.
- GROSS/klein beachten: Bei der Suche wird die Groß- und Kleinschreibung beachtet ("

 © 2007 Altova GmbH

Address " ist nicht gleich "address ").

- Regular expression: Suche nach Text, der durch die Regular Expression definiert wird, die Sie im Textfeld eingegeben haben. Eine Erläuterung zu Regular Expressions finden Sie unter Regular Expressions.

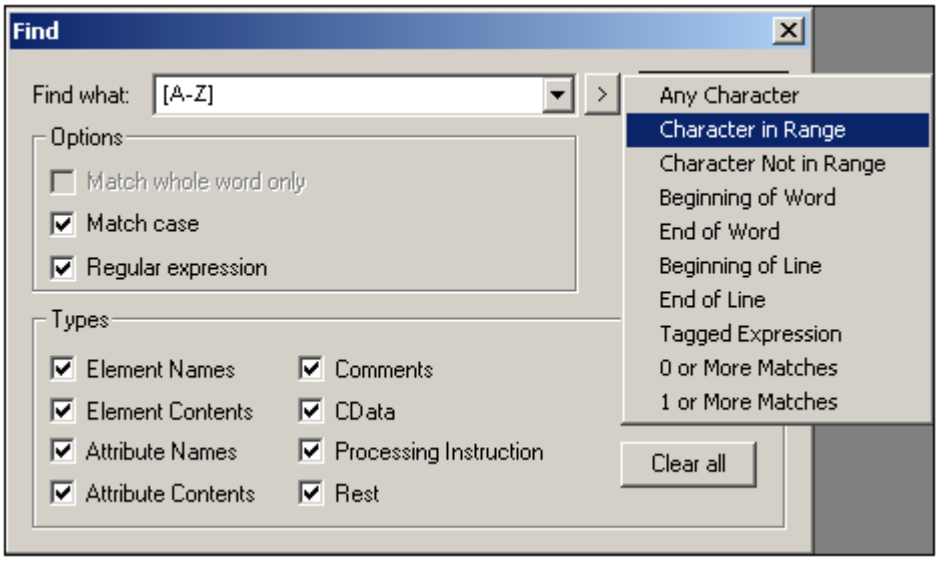
Schaltfläche "Erweitert": Öffnet Gruppe **Typ** (siehe Screenshot unten), wo Sie angeben können, welche XML-Typen durchsucht werden sollen.



Bitte beachten Sie:

- Das Suchdialogfeld hat keinen Modus, d.h. es kann geöffnet bleiben, während Sie in der Text-Ansicht arbeiten. Wenn Sie bei geöffnetem Dialogfeld die Eingabetaste drücken, wird es geschlossen. Wenn Text vor dem Öffnen des Dialogfelds markiert war, wird der markierte Text automatisch im Textfeld "Suchen nach:" übernommen.
- Wurde das Suchdialogfeld geschlossen, können Sie die aktuelle Suche wiederholen. Drücken Sie die **F3-**Taste um eine Suche nach unten bzw. **UMSCHALT+F3** um eine Suche nach oben durchzuführen.
- Durch Klicken auf die Erweiterungsschaltfläche rechts neben dem "Suchen nach:"-Listenfeld wird ein zweites Fenster geöffnet, über das Sie Regular Expressions eingeben können.

Regular Expressions:
Sie können Ihre Suchkriterien mit Hilfe von Regular Expressions näher definieren. Für die Erstellung der Regular Expression steht eine Popup-Liste zur Verfügung, die Sie über die >-Schaltfläche rechts vom Suchbegriff-Eingabefeld aufrufen können.



Wenn Sie auf die Beschreibung des gewünschten Suchausdrucks klicken, wird die entsprechende Expression-Syntax im Eingabefeld übernommen. Im Folgenden sehen Sie eine Liste von Syntax-Zeichen, die für Regular Expressions verwendet werden.

.

.	Steht für jedes beliebige Zeichen. Dies ist ein Platzhalter für ein einzelnes Zeichen.
\(	Markiert den Beginn einer Region, die gesucht werden soll.
\)	Markiert das Ende einer zu suchenden Region.
\n	n steht für 1 bis 9 und bezieht sich auf die erste bis neunte zu ersetzende Suchentsprechung. Lautet der Suchstring beispielsweise Fred\([1-9]\)XXX und der Ersetzungsstring Sam\1YYY, so wird Fred2XXX durch Sam2YYY ersetzt.
\<	Steht für den Beginn eines Worts.
\>	Steht für das Ende eines Worts.
\x	Damit können Sie ein Zeichen x verwenden, das sonst eine spezielle Bedeutung hätte. So würde z.B. \[als [und nicht als Beginn einer Zeichenmenge interpretiert werden.
[...]	Kennzeichnet eine Zeichenmenge, [abc] z.B. steht für jedes der Zeichen a, b oder c. Sie können auch Bereiche angeben, z.B. [a-z] für alle klein geschriebenen Zeichen.
[^...]	Die invertierte Zeichenmenge. [^A-Za-z] z.B. steht für jedes Zeichen mit Ausnahme alphabetischer Zeichen.
^	Steht für den Zeilenanfang (es sei denn dieses Zeichen wird innerhalb einer Menge verwendet, siehe oben).
$	Steht für das Zeilenende. Beispiel: A+$ findet ein oder mehrere A's am Ende der Zeile.
*	Steht für 0 oder öfter. Mit Sa*m werden z.B. Sm, Sam, Saam, Saaam usw. gefunden.
+	Steht für 1 oder öfter. Mit Sa+m werden z.B. Sam, Saam, Saaam usw. gefunden.

© 2007 Altova GmbH

15.2.14 Weitersuchen

F3

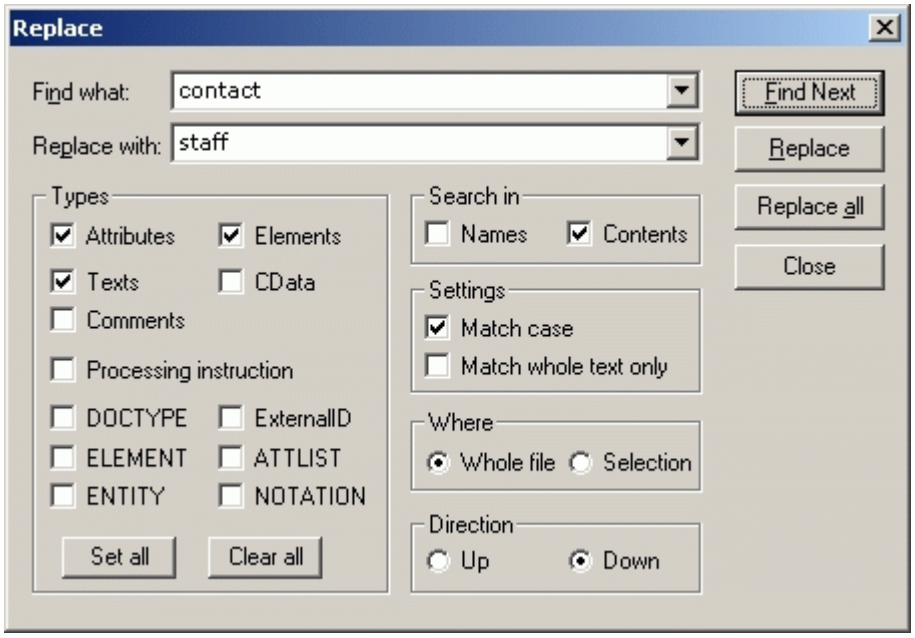
Mit **Weitersuchen** wird weiter nach dem zuvor eingegebenen Begriff gesucht.

15.2.15 Ersetzen...

Strg + H

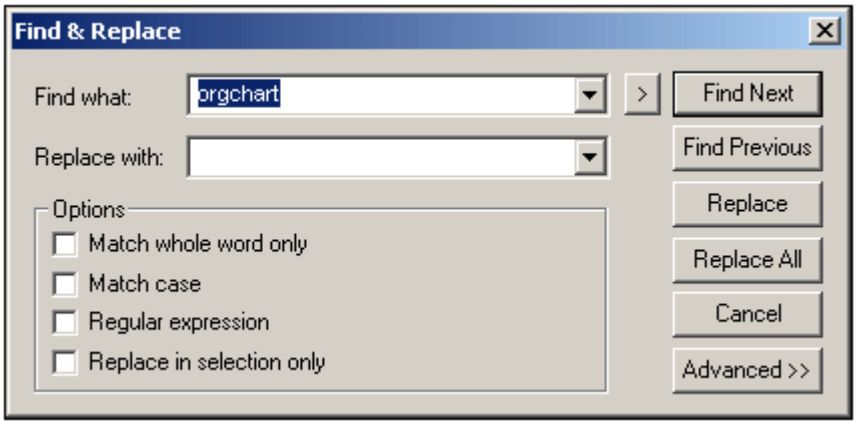
Mit dem Befehl **Ersetzen** können Sie einen Textstring durch einen anderen ersetzen. Der Befehl verfügt über dieselben Optionen wie der Befehl Suchen...
Abhängig von der ausgewählten Ansicht, gibt es im Dialogfeld "Ersetzen" verschiedene zur Auswahl stehende Optionen. Sie können diese Funktion Schritt für Schritt verwenden, oder alle Begriffe auf einmal ersetzen, indem Sie auf die Schaltfläche "Alle ersetzen" klicken.

Grid-Ansicht
In der unten gezeigten Abbildung sehen Sie die verschiedenen Suchoptionen.

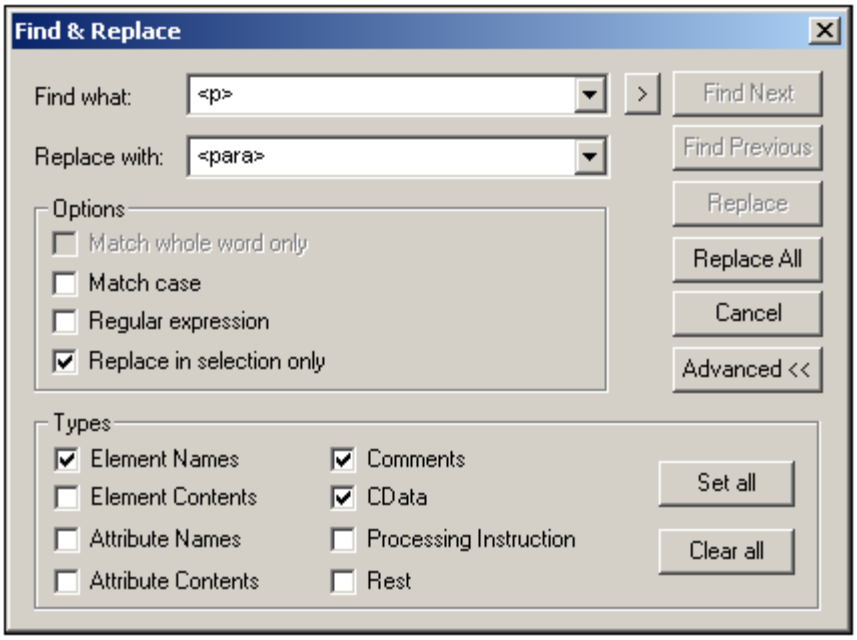


Diese Optionen sind im Abschnitt Suchen... beschrieben.

Text-Ansicht
Wenn Sie in der Text-Ansicht den Befehl "Ersetzen" wählen, wird das unten gezeigte Dialogfeld "Suchen & Ersetzen" angezeigt. Die Optionen sind dieselben wie im Dialogfeld Suchen.... Bei Auswahl der Option "Nur im markierten Bereich ersetzen" werden die Ersetzungen nur im markierten Dokumentbereich vorgenommen.



Wenn Sie auf die Schaltfläche "Erweitert" klicken, werden die Typ-Optionen angezeigt (nähere Informationen dazu siehe Suchen...).

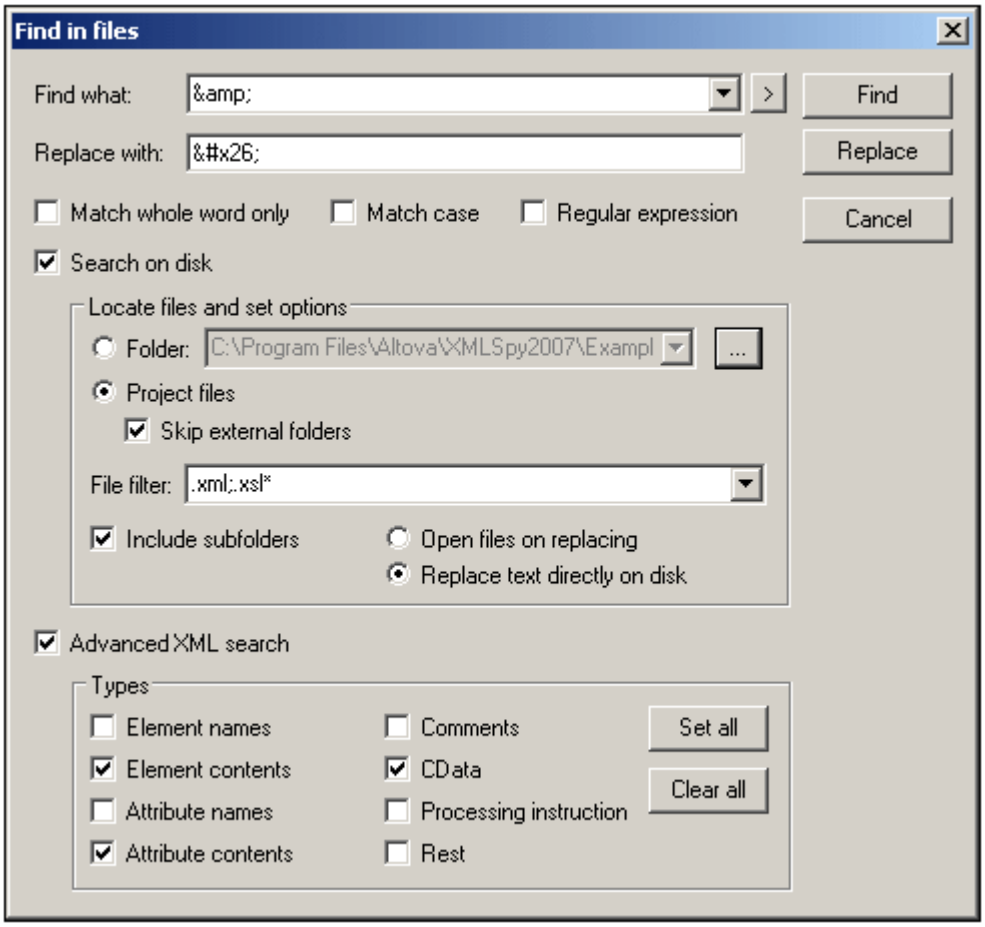


Bitte beachten Sie: Bei Auswahl des Befehls "Alle ersetzen" wird jede einzelne Ersetzung als Einzelvorgang aufgezeichnet, sodass Sie diesen Vorgang Schritt für Schritt rückgängig machen können.

15.2.16 In Dateien suchen

Das Dialogfeld **In Dateien suchen** ist eine ideale Methode, um Text schnell in vielen Dateien gleichzeitig zu suchen und zu ersetzen. Wenn Sie den Befehl aufrufen, wird das Dialogfeld "In Dateien suchen" (*Abbildung unten*) angezeigt. Der Befehl **In Dateien suchen** unterscheidet sich insofern vom Befehl **Suchen** als damit alle angegebenen Dateien auf einmal nach dem gewünschten Suchstring durchsucht werden und der Ersetzungsvorgang überall gleichzeitig durchgeführt wird. Anschließend wird in der "In Dateien suchen"-Ausgabeleiste ein Bericht angezeigt. Bei Verwendung des Befehls **Suchen** gibt der Benutzer den Suchstring ein und geht ein einzelnes Dokument (das aktive Dokument) von Instanz zu Instanz durch.

 © 2007 Altova GmbH



Suchkriterien
Es gibt zwei grundlegende Suchkriterien: (i) Was soll gesucht werden und (ii) wo soll gesucht werden? Eine Anleitung, wie man den zu suchenden Text eingibt (was soll gesucht werden), finden Sie in der Beschreibung zum Befehl **Suchen**. Wenn es sich bei dem im Textfeld "Suchen nach" eingegebenen Text um eine Regular Expression handelt, muss das Kontrollkästchen "Regular Expression" aktiviert werden. Um eine Eingabehilfe zur Eingabe von Regular Expressions aufzurufen, klicken Sie auf die Schaltfläche . Die Verwendung von Regular Expressions zur Suche wird unter Suchen erklärt.

Um festzulegen, welche Node-Typen und Teile eines XML-Dokuments durchsucht werden sollen, aktivieren Sie das Kontrollkästchen "Erweiterte XML-Suche" und aktivieren Sie anschließend die gewünschten Node-Typen.

Sie können festlegen, welche Dateien durchsucht werden, indem Sie den gewünschten Ordner oder das gewünschte Projekt angeben (nachdem Sie das Kontrollkästchen "Auf Festplatte suchen" aktiviert haben). Bei Auswahl eines Projektordners können externe Ordner, die zu einem Projekt hinzugefügt wurden, übersprungen werden. Die zu durchsuchenden Dateien können nach Dateierweiterung gefiltert werden. Das Trennzeichen zwischen zwei Dateierweiterungen kann ein Komma oder ein Semikolon sein. Das Stern-Zeichen kann ebenfalls als Wildcard verwendet werden.

Die Instanzen, die vom Suchstring in allen angegebenen Suchordnern gefunden werden, werden in der Ausgabeleiste "In Dateien suchen" aufgelistet. Wenn Sie auf einen der aufgelisteten Einträge klicken, wird die entsprechende Datei in der Text-Ansicht geöffnet und

die Instanz wird markiert.

Ersetzen
Beachten Sie: Wenn Sie auf die Schaltfläche **Ersetzen** klicken, werden alle Instanzen des Suchstrings durch den Ersetzungsstring ersetzt. Wenn Sie im Dialogfeld "In Dateien suchen" die Option "Dateien beim Ersetzen öffnen" aktiviert haben, wird die Datei in der Text-Ansicht geöffnet. Andernfalls geschieht die Ersetzung im Hintergrund. Alle ersetzten Strings werden in der In Dateien suchen-Ausgabeleiste aufgelistet. Wenn Sie auf einen der aufgelisteten Einträge klicken, wird die Datei in der Text-Ansicht geöffnet und der Eintrag wird markiert.

15.2.17 Lesezeichen einfügen/löschen

Strg+F2

Mit dem Befehl **Lesezeichen einfügen/löschen** können Sie ein Lesezeichen an der aktuellen Cursorposition einfügen, bzw. ein vorhandenes Lesezeichen entfernen, wenn sich der Cursor in einer Zeile befindet, in der zuvor ein Lesezeichen eingefügt wurde. Dieser Befehl steht nur in der Textansicht zur Verfügung.

Mit Lesezeichen markierte Zeilen können auf zwei Arten angezeigt werden:

- Wurde die Lesezeichenleiste aktiviert, wird eine blaue Ellipse links vom Text in der Lesezeichenleiste angezeigt.
- Ist die Lesezeichenleiste nicht aktiviert, so wird die gesamte Zeile, in der sich der Cursor befindet markiert.

Mit der **F2**-Taste können Sie der Reihe nach alle Lesezeichen in der Textansicht anzeigen.

15.2.18 Alle Lesezeichen löschen

Strg+Umschalt+F2

Mit diesem Befehl löschen Sie alle derzeit in der Textansicht definierten Lesezeichen. Dieser Befehl steht nur in der Textansicht zur Verfügung.

Bitte beachten Sie: Die Auswirkungen dieses Befehls können mit dem Befehl "Rückgängig" nicht rückgängig gemacht werden.

15.2.19 Zum nächsten Lesezeichen

F2

Setzt den Cursor an den Beginn der nächsten mit einem Lesezeichen versehenen Zeile. Dieser Befehl ist nur in der Textansicht verfügbar.

15.2.20 Zum vorhergehenden Lesezeichen

Umschalt+F2

Setzt den Textcursor an den Beginn der vorhergehenden mit einem Lesezeichen versehenen Zeile. Dieser Befehl ist nur in der Textansicht verfügbar.

© 2007 Altova GmbH

15.3 Menü "Projekt"

XMLSpy verwendet die bekannte Baumstruktur, um Dateien und URLs in einem XML-Projekt zu gliedern. Dateien und URLs mit gleicher Erweiterung können in Ordnern gesammelt werden, wodurch eine einfache Strukturierung und Batchmanipulation gegeben ist.

Bitte beachten Sie:
Die meisten projektbezogenen Befehle sind auch mittels Rechtsklick auf das gewünschte Objekt über das Kontextmenü verfügbar.

Absolute und relative Pfade
Jedes Projekt ist als Projektdatei gespeichert, die mit der Erweiterung **.spp** versehen ist. Bei diesen Dateien handelt es sich eigentlich um XML-Dokumente, die Sie wie jede reguläre XML-Datei editieren können. In der Projektdatei werden für Dateien/Ordner auf derselben oder einer höheren Ebene absolute Pfade verwendet, während für Dateien/Ordner im aktuellen Ordner oder einem Unterordner relative Pfade definiert werden. Wenn Ihre Verzeichnisstruktur z.B. in etwa so aussieht:

```
|-Folder1
|      |
|      |-Folder2
|             |
|             |-Folder3
|                    |
|                    |-Folder4
```

...und Ihre **.spp**-Datei in `Folder3` gespeichert ist, so werden Referenzen auf Dateien in `Folder1` und `Folder2` folgendermaßen angegeben:

```
c:\Folder1\NameOfFile.ext
c:\Folder1\Folder2\NameOfFile.ext
```

Referenzen auf Dateien in `Folder3` und `Folder4` würden in etwa folgendermaßen aussehen:

```
.\NameOfFile.ext
.\Folder4\NameOfFile.ext
```

Wenn Sie sicherstellen möchten, dass alle Pfade relativ angegeben werden, speichern Sie die `.spp`-Dateien im Root-Verzeichnis Ihres Rechners.

Projekte und Versionskontrolle
Wenn Sie vorhaben, ein XMLSpy-Projekt zu einem Versionskontroll-Repository hinzuzufügen, stellen Sie bitte sicher, dass die Projektdateien in der hierarchischen Dateisystemstruktur in einem Ordner abgelegt werden, von dem aus Dateien nur aus diesem untergeordneten Ordnern hinzugefügt werden können (Nehmen Sie das Root-Verzeichnis als oberste Ebene der Verzeichnisstruktur)

In anderen Worten stellt das Verzeichnis, in dem sich die **Projektdatei** befindet, innerhalb des Versionskontroll-Repository im Grunde das **Root-Verzeichnis** des Projekts dar. Dateien, die dem Projekt aus einem Ordner hinzugefügt werden, der sich oberhalb dieses Root-Verzeichnisses befindet, werden dem XMLSpy-Projekt zwar hinzugefügt, doch ist ihr Speicherort im Repository - falls überhaupt zulässig - manchmal ein unerwarteter.

Nehmen wir als Beispiel die oben gezeigte Verzeichnisstruktur:
Wenn eine Projektdatei in `Folder3` gespeichert wird und einer Versionskontrolle unterliegt:

- so werden Dateien, die zu `Folder1` hinzugefügt werden, unter Umständen nicht von der Versionskontrolle berücksichtigt,

 © 2007 Altova GmbH

- werden Dateien, die zu `Folder2` hinzugefügt werden, dem Root-Verzeichnis des Repository anstelle des Projektordners hinzugefügt, jedoch von der Versionskontrolle berücksichtigt,
- während die in `Folder3` und `Folder4` gespeicherten Dateien korrekt verarbeitet werden und auch der Versionskontrolle unterliegen.

15.3.1 Neu

Mit diesem Befehl erstellen Sie ein **neues** Projekt in XMLSpy.
Wenn Sie gerade an einem anderen Projekt arbeiten, werden Sie gefragt, ob Sie alle Dokumente schließen wollen, die zu dem aktuellen Projekt gehören.

15.3.2 Öffnen...

Diese Funktion **öffnet** ein bereits vorhandenes Projekt inXMLSpy.
Falls Sie zuvor an einem anderen Projekt gearbeitet haben, wird dieses Projekt zuerst geschlossen.

15.3.3 Neu laden

Mit dieser Funktion wird das aktuelle Projekt von der Festplatte **neu geladen**.
Falls Sie in einer Multiuser-Umgebung arbeiten, kann es oftmals nötig sein, ein Projekt, an dem mehrere Personen gleichzeitig arbeiten, neu zu laden, um deren Änderungen anzusehen.

Bitte beachten Sie: Projektdateien (.spp) sind XML-Dokumente, die Sie wie jede andere XML Datei bearbeiten können.

15.3.4 Schließen

Mit diesem Befehl schließen Sie das **aktuelle** Projekt. Wurde das Projekt geändert, werden Sie vor dem Schließen gefragt, ob die Änderungen gespeichert werden sollen.
Wenn ein Projekt oder eine XML-Datei geändert wurde, wird der Name der Datei in der Titelleiste des Projekts mit einem Stern "*" versehen.

15.3.5 Speichern

Mit diesem Befehl wird das aktuelle Projekt **gespeichert**. Sie können ein Projekt auch speichern, indem Sie das Projektfenster zum aktiven machen und auf das Symbol klicken.

15.3.6 Versionskontrolle

XMLSpy unterstützt Microsoft Source-Safe und andere kompatible Speicherschnittstellen.

Open Source Versionskontrollsysteme:

- Concurrent Versions System (CVS).
  Um XMLSpy mit CVS verbinden zu können, müssen Sie die Software Jalindi Igloo herunterladen. Sie finden diese Softwarekomponente entweder auf der Altova -Komponenten-Website im Abschnitt Komponenten oder auf der Website von Jalindi.com .

Kommerzielle Versionskontrollsysteme:

- Microsoft Visual Source-Safe 6.0c
- PVCS Version Manager *
- StarTeam 5.1 *

* Die Kompatibilität mit diesem Drittanbieter-Tool ist von der Implementierung der SCC-Schnittstelle abhängig und kann nicht ausdrücklich garantiert werden.

Ein Versionskontrollprojekt ist allerdings nicht mit einem Projekt in XMLSpy zu vergleichen. Versionskontrollprojekte sind verzeichnisbezogen, wobei XMLSpy Projekte logische Konstrukte ohne direkte Verzeichnisabhängigkeit sind.

Im folgenden Beispiel wird ein Text verwendet, der Microsoft Source-Safe als Versionskontrolle benutzt.

Bitte beachten Sie:
Microsoft hat einen Registryeintrag definiert, bei dem alle SCC-kompatiblen Programme sich selbst registrieren. XMLSpy liest nur diesen Eintrag.

HKEY_LOCAL_MACHINE\SOFTWARE\SourceCodeControlProvider\InstalledSCCProviders

Delta V
Erweitert das Protokoll WebDAV, wobei WebDAV eine Erweiterung des Protokolls HTTP 1.1 ist. XMLSpy unterstützt Delta V auf Projekt,- Verzeichnis-, Ordner- und Dateiebene über das Kontextmenü, das mit Rechtsklick auf das gewünschte Objekt aufgerufen wird. Das DeltaV Protokoll wird von XMLSpy unterstützt, wird aber nur verwendet, wenn der Server, auf dem die Dateien unter Versionskontrolle stehen, dies zulässt.

Beim Zugriff auf Dateien, die auf einem Server abgelegt sind, der DeltaV unterstützt, werden automatisch die Funktionen Einchecken/Auschecken/Auschecken rückgängig aktiviert.

Delta V Befehle werden durch Rechtsklick auf folgende Objekte aufgerufen:

- Dateiregister einer offenen Datei im Hauptfenster
- Datei im Fenster "URL öffnen"
- Datei in einem externen Webverzeichnis eines Projekts

Delta V ist der **Versionierungsteil** des WebDAV Standards, und unterstützt:

- Auschecken
- Auschecken rückgängig
- Einchecken (inkl. Kommentare)

WebDAV Merkmale:

- Eigenschaften: Auflisten, hinzufügen, löschen

© 2007 Altova GmbH

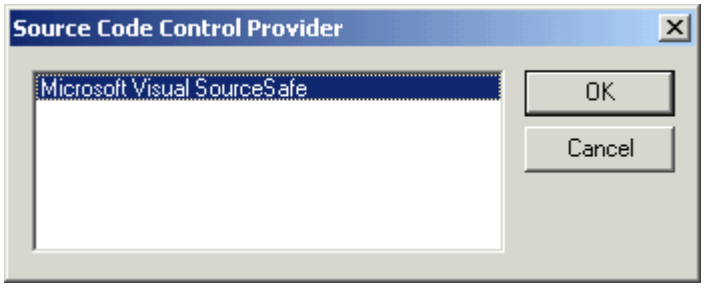
- Namespace Aktionen: Verschieben und Kopieren
- Sammlungen: Hierarchieoperationen und mkcol.

Weitere Informationen über DeltaV finden Sie unter http://www.webdav.org/deltav.

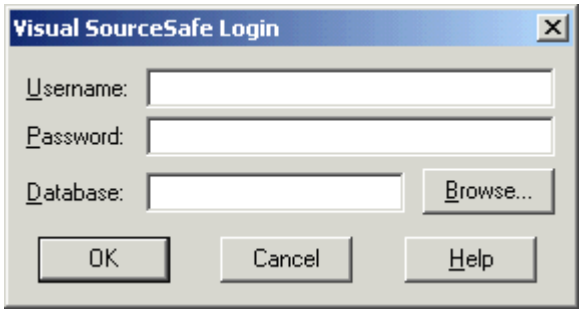
Öffnen

Mit der Funktion "Öffnen" wird ein bereits vorhandenes Versionskontrollprojekt lokal wiedergegeben (oder ein neues Projekt erstellt). Diese Funktion kann nur ausgeführt werden, wenn ein XMLSpy Projekt zuvor in die Versionskontrolle mittels der Menüoption Zu Versionskontrolle hinzufügen übernommen wurde.

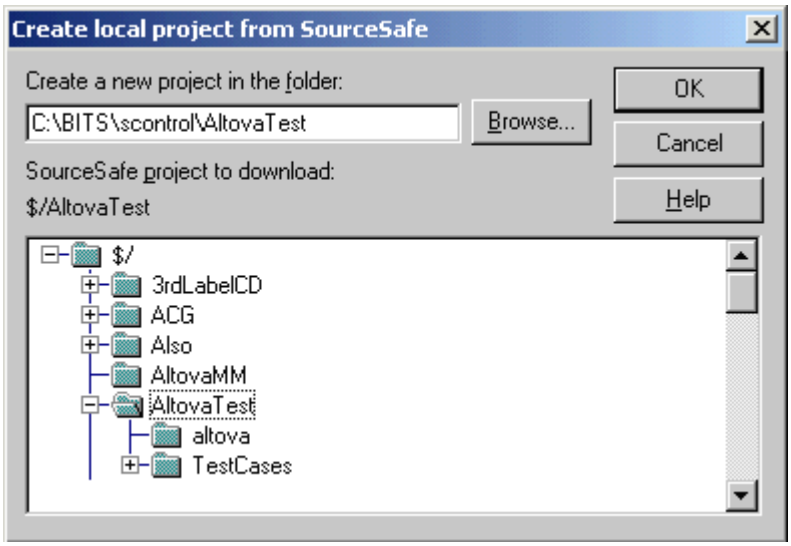
1. Wählen Sie **Projekt** | **Versionskontrolle** | **Öffnen**.
2. Wählen Sie eine Versionskontrolle, die auf Ihrem PC installiert ist, und bestätigen Sie mit der Schaltfläche OK.



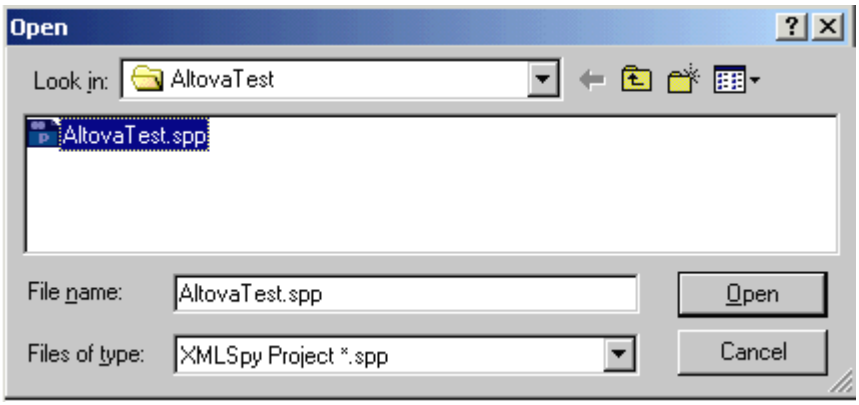
3. Geben Sie Ihre Login-Daten im Versionskontrollfenster ein, und bestätigen Sie mit OK. Daraufhin wird das Dialogfeld zum Erstellen eines lokalen Projekts über die Versionskontrolle angezeigt.



4. Geben Sie an, in welchem Verzeichnis das lokale Projekt abgelegt werden soll.
5. Wählen Sie das herunterzuladende Projekt.
   Falls das gewählte Verzeichnis noch nicht existiert, werden Sie aufgefordert, es zu erstellen. Klicken Sie auf "Ja" zum Erstellen des neuen Ordners.
6. Klicken Sie auf die Schaltfläche OK um das Versionskontrollprojekt herunterzuladen.



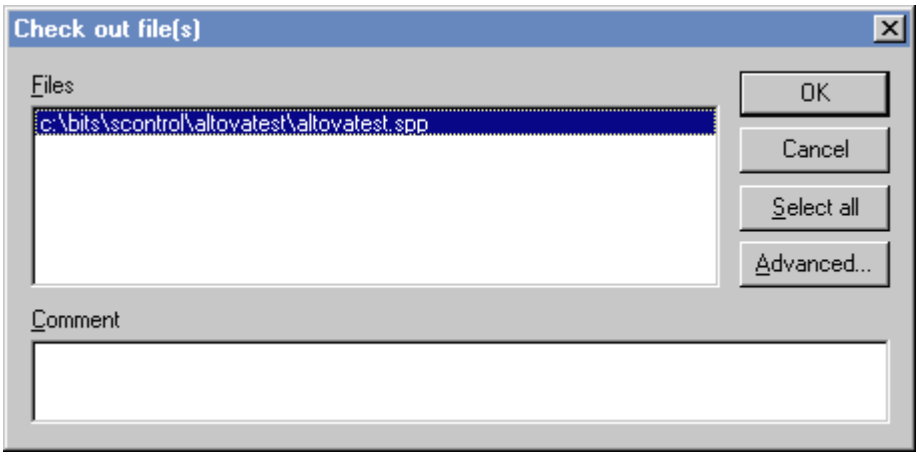
7. Das Dialogfeld "Öffnen" wird angezeigt. Hier können Sie die XMLSpy Projektdatei (*.spp) im lokalen Verzeichnis suchen. Klicken Sie auf die zu öffnende Projektdatei.



Falls sich das Projekt bereits unter Versionskontrolle befindet, erhalten Sie eine Meldung, dass das Projekt unter Versionskontrolle steht. Sie können nun entscheiden, ob Sie das Projekt auschecken möchten.

8. Klicken Sie auf die Schaltfläche OK um die Versionskontrolle zu aktivieren. Das Dialogfeld Auschecken wird geöffnet. Nun können Sie Ihre *.spp Datei auschecken.
9. Im Textfeld **Kommentare** können Sie Kommentare zu Dateien eingeben. Klicken Sie auf OK um die XMLSpy Projektdatei auszuchecken. Je nach Versionskontrollsystem enthält das Dialogfeld unter Umständen weitere Befehlsschaltflächen. Mit der Schaltfläche **Erweitert** öffnen Sie das Dialogfeld "Erweiterte Checkout-Funktionen".

© 2007 Altova GmbH



Die *.spp Datei ist nun ausgecheckt, und alle XML Dateien, die im XMLSpy Projekt enthalten sind, werden in das lokale Verzeichnis kopiert.

Resultat in XMLSpy:
Das AltovaTest Verzeichnis wurde lokal erstellt. Alle Dateien des XMLSpy Projekts werden nun im Projektfenster angezeigt. Sie können nun mit diesen Dateien arbeiten.

Symbole der Versionskontrolle:

Das **rote Kontrollhäkchen** zeigt an, dass diese Datei **ausgecheckt** ist, z.B.ist die Projektdatei "AltovaTest" ausgecheckt.
Das Sternchen am Ende des Dateinamens bedeutet, dass diese Datei verändert wurde. Beim Schließen werden Sie aufgefordert, die Datei zu speichern.

Das Vorhängeschloss-Symbol oben rechts im XML Symbol zeigt an, dass diese Datei **unter Versionskontrolle steht**, aber derzeit nicht ausgecheckt ist.

Dieses URL-Dokument ist Teil des AltovaTest Projektes, steht aber **nicht unter Versionskontrolle**. Dokumente, die über eine URL geöffnet wurden, können nicht unter Versionskontrolle gestellt werden.

Versionskontrolle aktivieren

Diese Funktion erlaubt es, die Versionskontrolle für ein Projekt zu aktivieren oder deaktivieren.

So deaktivieren Sie die Versionskontrolle für ein Projekt:

1. Klicken Sie auf die Projektdatei oder eine beliebige Datei im Projektfenster.
2. Wählen Sie die Menüoption **Projekt | Versionkontrolle** und deaktivieren Sie das Kontrollkästchen **Versionskontrolle aktivieren**.

Um die Versionskontrolle **provisorisch** zu deaktivieren, wählen Sie Ja.

Um die Versionskontrolle für dieses Projekt **permanent** zu deaktivieren, wählen Sie Nein.

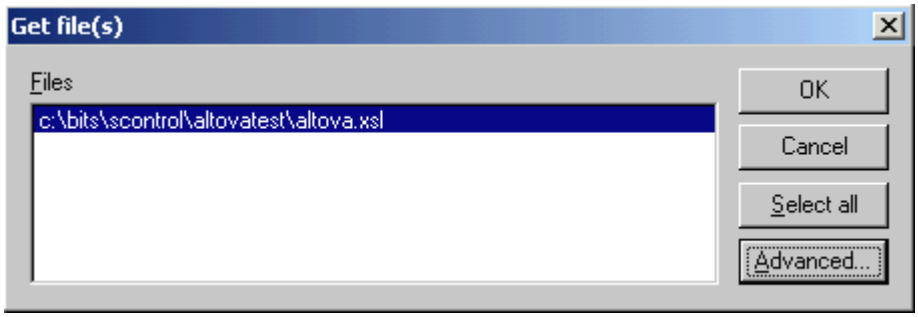
Aktuellste Version holen

Mit diesem Befehl wird die aktuellste Version der ausgewählten Datei geholt.

So rufen Sie die neueste Version einer Datei ab:

- Wählen Sie **Projekt** | **Versionskontrolle** | **Aktuellste Version holen**.

Diese Datei ist eine Kopie der Datei unter Versionskontrolle und wird in der lokalen Arbeitsmappe Ihres Computers abgelegt.



Auschecken

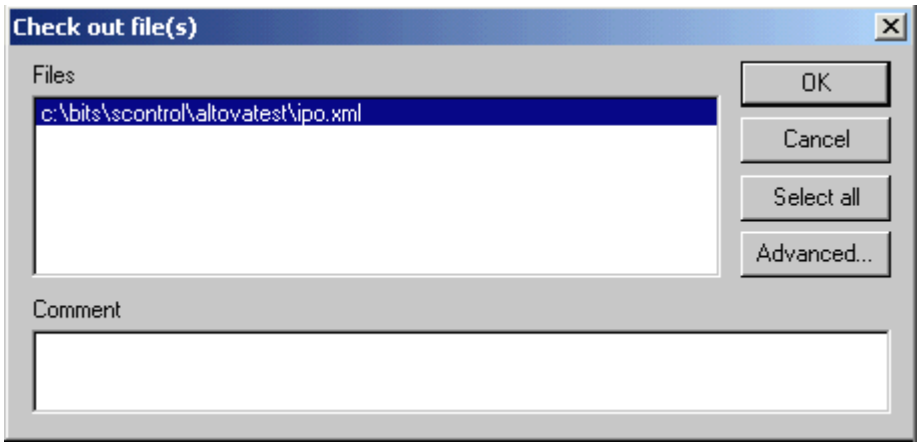
Mit diesem Befehl checken Sie die aktuellste Version der ausgewählten Datei(en) aus. Die Dateien werden für andere Benutzer als "ausgecheckt" markiert.

So checken Sie Dateien aus:

- Wählen Sie **Projekt** | **Versionskontrolle** | **Auschecken**.

Kürzel: Rechtsklicken Sie auf ein Objekt im Projektfenster, und wählen Sie "Auschecken" im Kontextmenü.

© 2007 Altova GmbH



Folgende Objekte können ausgecheckt werden:

- Einzelne Dateien - klicken Sie auf die Datei (oder mehrere Dateien mittels Strg + Klick)
- XMLSpy Projektordner - klicken Sie auf den Ordner (oder mehrere Ordner mittels Strg + Klick)
- XMLSpy Projekt - klicken Sie auf das Symbol der Projektdatei.

Das rote Kontrollhäkchen zeigt an, dass die Datei ausgecheckt ist.

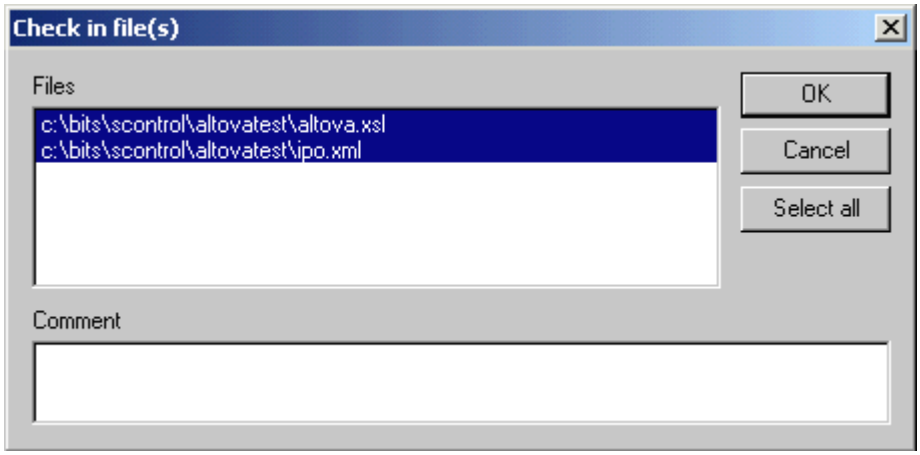
Einchecken

Mit diesem Befehl checken Sie zuvor ausgecheckte Dateien wieder ein (z.B. lokal aktualisierte Dateien), und stellen die Dateien zurück in das Versionskontrollprojekt.

So checken Sie Dateien ein:

- Wählen Sie **Projekt** | **Versionskontrolle** | **Einchecken**.

Kürzel: Rechtsklicken Sie auf ein ausgechecktes Objekt, und wählen Sie "Einchecken" im Kontextmenü.



Folgende Objekte können eingecheckt werden:

- Einzelne Dateien - klicken Sie auf die Datei (oder mehrere Dateien mittels Strg + Klick)
- XMLSpy Projektordner - klicken Sie auf den Ordner (odere mehrere Ordner mittels Strg

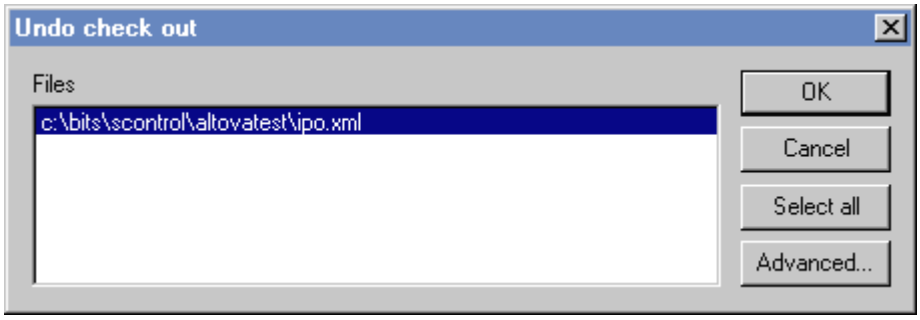
+ Klick)
- Ein komplettes XMLSpy Projekt - klicken Sie auf das Symbol der Projektdatei.

Das Vorhängeschloss-Symbol zeigt an, dass sich die Datei **unter Versionskontrolle** befindet, aber derzeit nicht ausgecheckt ist.

Auschecken rückgängig...

Mit diesem Befehl werden Änderungen in zuvor ausgecheckten Dateien **verworfen**. Lokal aktualisierte Dateien können gegen ältere Dateien im Versionskontrollprojekt ausgetauscht werden.

Kürzel: Rechtsklicken Sie auf ein Objekt im Projektfenster, und wählen Sie "Auschecken rückgängig" im Kontextmenü.

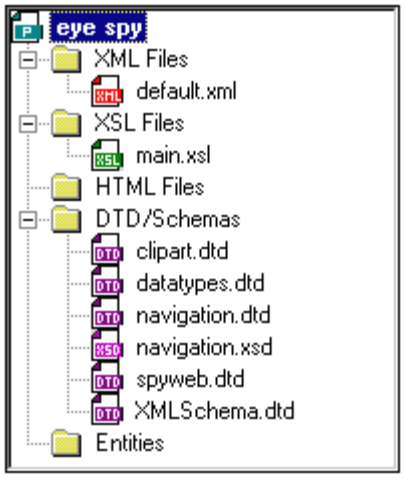


Auschecken rückgängig kann auf folgende Objekte angewendet werden.
- Einzelne Dateien - klicken Sie auf die Datei (oder mehrere Dateien mittels Strg + Klick)
- XMLSpy Projektordner - klicken Sie auf den Ordner (odere mehrere Ordner mittels Strg + Klick)
- XMLSpy Projekt - klicken Sie auf das Symbol der Projektdatei.

© 2007 Altova GmbH

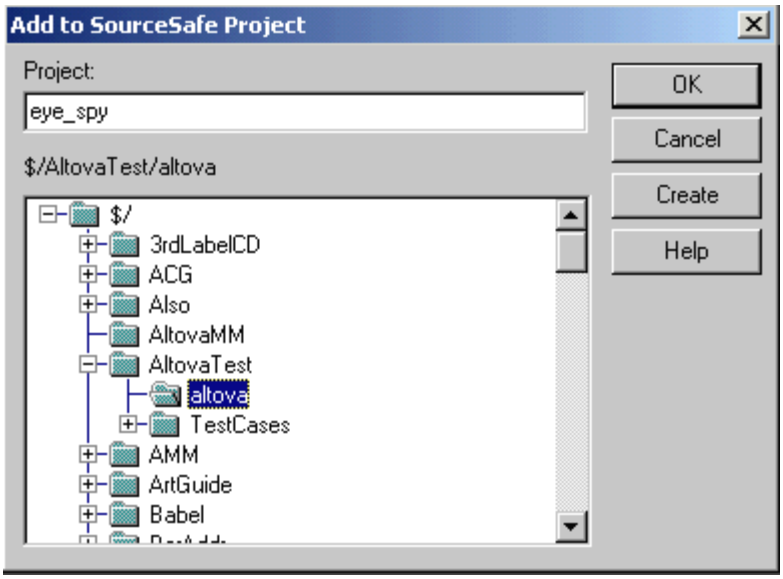
Zu Versionskontrolle hinzufügen

Dieser Befehl fügt ein XMLSpy Projekt in Ihre Versionskontrolle ein.

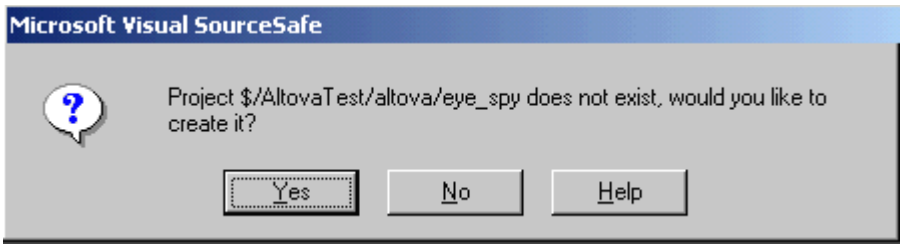


So fügen Sie ein XML Projekt einer Versionskontrolle hinzu:

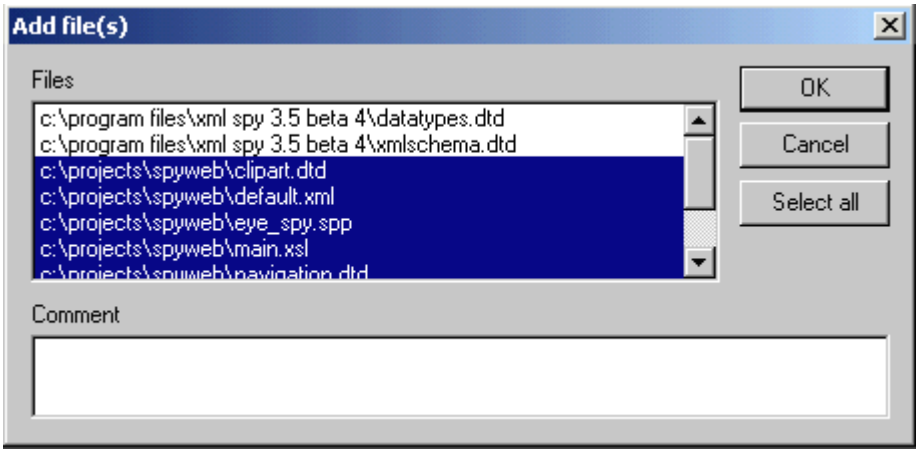
1. Klicken Sie auf das XMLSpy Projektsymbol, und wählen Sie die Menüoption **Projekt | Versionskontrolle | Zu Versionskontrolle hinzufügen.**
2. Wählen Sie Ihr Versionkontrollprodukt aus der Liste.
3. Geben Sie die benötigten Login-Daten ein.
4. Wählen Sie das Versionskontrollprojekt (Verzeichnis), in das Ihr XML Projekt eingefügt werden soll, und bestätigen sie mit OK.



Falls das Projekt nicht existiert, öffnet sich folgendes Dialogfeld:

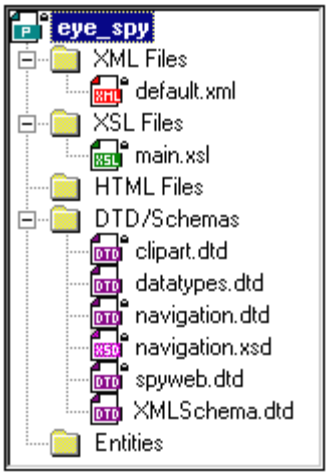


5. Wählen Sie Ja um ein neues Projekt zu erstellen. Mit Nein brechen Sie diesen Vorgang ab.



6. Wählen Sie die XMLSpy Projektdateien, die Sie unter Versionskontrolle stellen möchten aus, und bestätigen Sie mit OK.

Diese Dateien werden nun in die Versionskontrolle eingefügt, und befinden sich unter Versionskontrolle. Die Dateien sind durch das Schloss-Symbol gekennzeichnet. Nicht transferierte Dateien haben kein Schloss-Symbol (in der unteren Abbildung: datatypes.dtd und XMLschema.dtd).



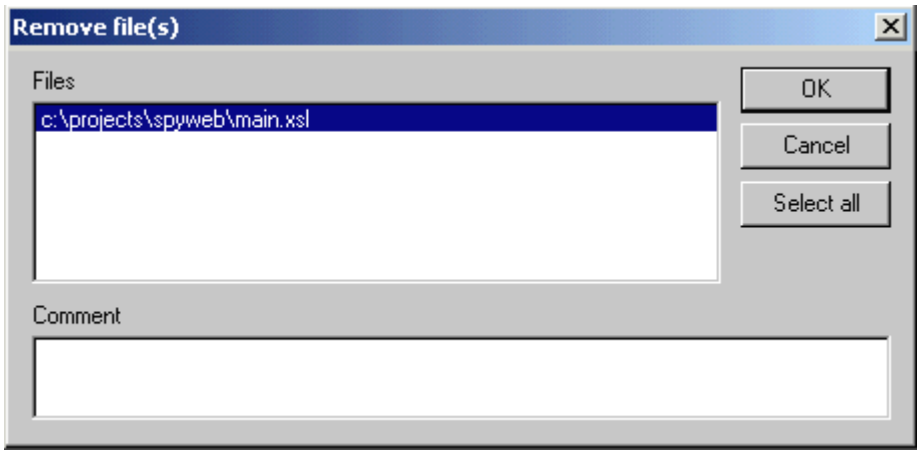
Von Versionskontrolle ausgliedern

Mit diesem Befehl **löschen** Sie zuvor hinzugefügte Dateien aus dem Versionskontrollprodukt.

 © 2007 Altova GmbH

So entfernen Sie Dateien aus der Versionskontrolle:

- Wählen Sie **Projekt** | **Versionskontrolle** | **Von Versionskontrolle ausgliedern**.



Folgende Objekte könne aus der Versionskontrolle ausgegliedert werden:

- Einzelne Dateien - klicken Sie auf die Datei (oder mehrere Dateien mittels Strg + Klick)
- XMLSpy Projektordner - klicken Sie auf den Ordner (oder mehrere Ordner mittels Strg + Klick)
- XMLSpy Projekt - klicken Sie auf das Symbol der Projektdatei.

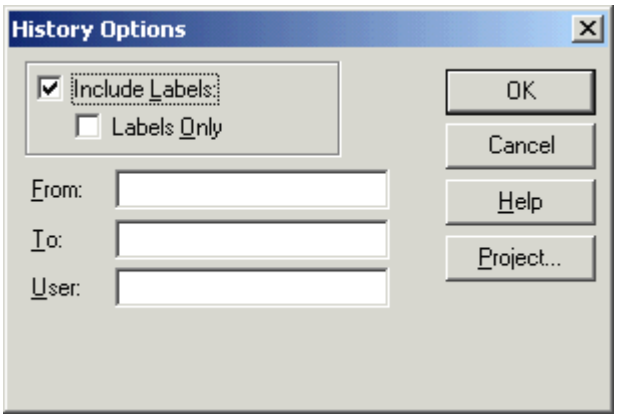
Verlauf anzeigen

Mit diesem Befehl zeigen Sie den Verlauf einer bestimmten Datei an. Der Verlauf kann jeweils nur für eine Datei angezeigt werden.

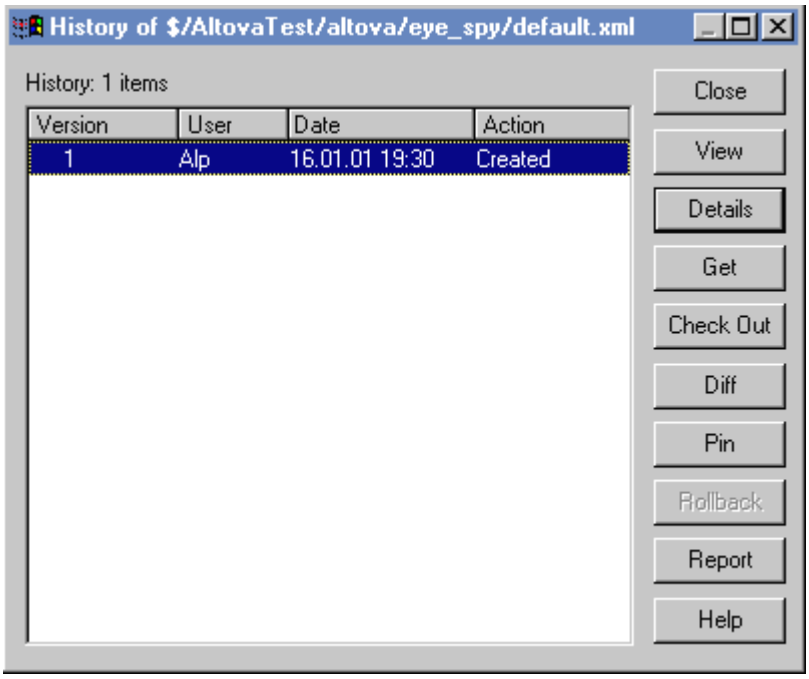
So zeigen Sie den Verlauf einer Datei an:

1. Klicken Sie auf die Datei im Projektfenster, deren Verlauf Sie ansehen möchten.
2. Wählen Sie die Menüoption **Projekt | Versionskontrolle | Verlauf anzeigen**.

   Im folgenden Dialogfeld können Sie weitere Informationen eingeben (in diesem Beispiel handelt es sich um ein Dialogfeld von Microsoft Source-Safe).



3. Wählen Sie den gewünschten Eintrag und bestätigen Sie mit OK.

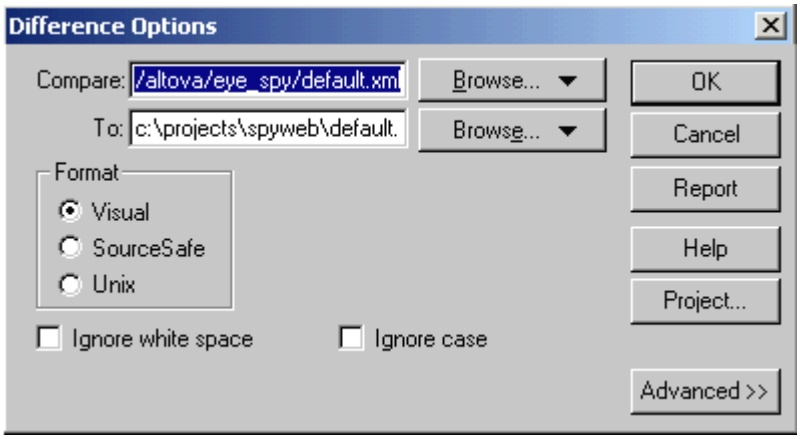


Unterschiede anzeigen

Dieser Befehl ermöglicht die Anzeige von Unterschieden zwischen der in der Versionskontroll-Ablage befindlichen und der gleichnamigen Datei, die Sie ausgecheckt haben. Dieser Befehl kann jeweils nur auf eine einzelne Datei angewendet werden.

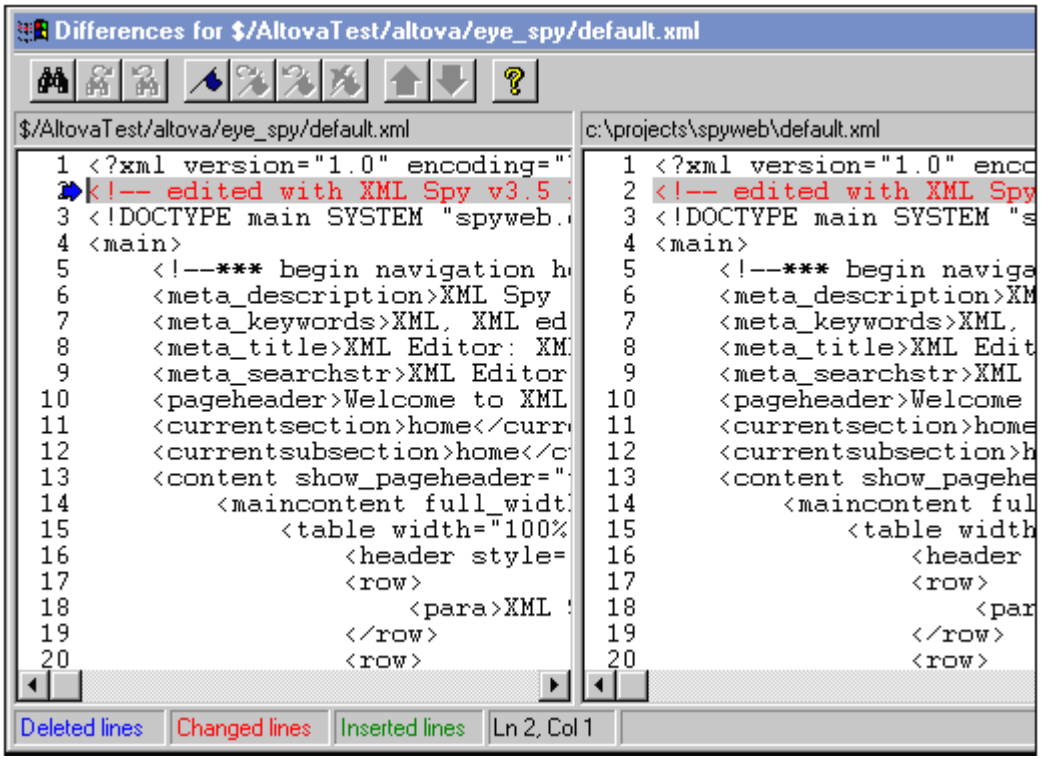
So zeigen Sie die Unterschiede zweier Dateien an:

1. Klicken Sie auf die Datei im Projektfenster.
2. Wählen Sie die Menüoption **Projekt | Versionskontrolle | Unterschiede anzeigen.** Ein Dialogfeld kann an dieser Stelle erscheinen, in dem Sie weitere Informationen eingeben können.



3. Wählen Sie die gewünschten Einträge und bestätigen Sie mit OK.

 © 2007 Altova GmbH



Die Unterschiede zwischen beiden Dateien werden in beiden Fenstern farblich markiert (dieses Beispiel verwendet MS Source-Safe).

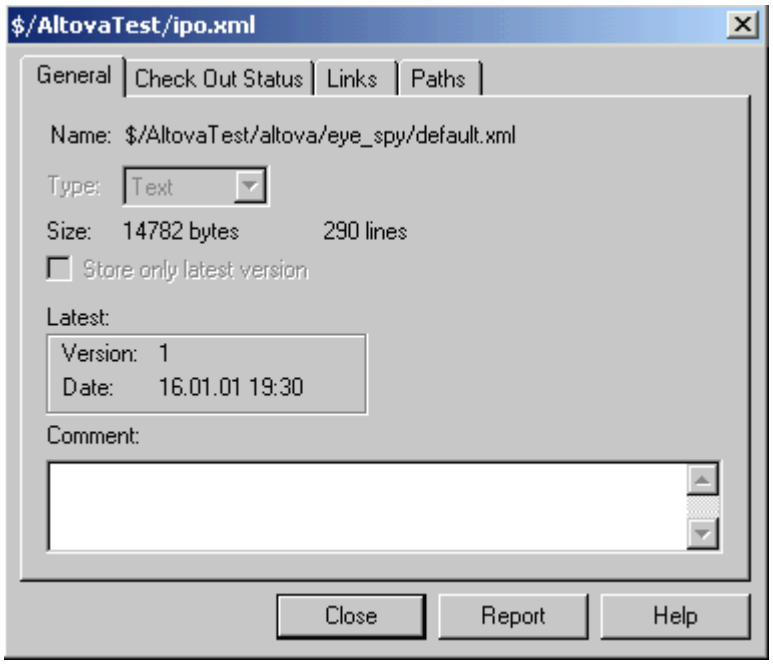
Eigenschaften

Mit diesem Befehl können Sie die Eigenschaften der ausgewählten Datei anzeigen. Die Anzeige kann je nach verwendetem Versionskontrollprodukt unterschiedlich sein.

So zeigen Sie die Eigenschaften der aktuell ausgewählten Datei an:

- Wählen Sie **Projekt** | **Versionskontrolle** | **Eigenschaften**.

Dieser Befehl kann jeweils nur an einer einzelnen Datei ausgeführt werden.



Status aktualisieren

Mit diesem Befehl **aktualisieren** Sie den Status aller Projektdateien unabhängig von ihrem derzeitigen Status.

Native Oberfläche ausführen

Mit diesem Befehl **starten** Sie die ursprüngliche Oberfläche Ihrer Versionskontroll-Software.

15.3.7 Dateien hinzufügen...

Mit diesem Befehl fügen Sie Dateien in das aktuelle Projekt ein.Verwenden Sie diesen Befehl um Dateien in einen beliebigen Ordner Ihres Projekts einzufügen. Sie können einzelne Dateien oder Dateigruppen (Auswahl mit **Strg + Klick**) im Dialogfeld "Öffnen" hinzufügen. Wenn Sie Dateien in ein Projekt einfügen, werden diese automatisch - nach Erweiterung sortiert - in die entsprechenden Projektordner verteilt (Einstellungen hierzu im Dialogfeld Projekteigenschaften).

15.3.8 URL hinzufügen...

Mit diesem Befehl fügen Sie eine URL in das aktuelle Projekt ein. Bei Verwendung von URLs in einem Projekt wird das Ziel der URL in das Projekt einbezogen. Wenn eine Batch-Operation an einer URL oder einer Datei mit URL ausgeführt wird, sucht XMLSpy nach dem Dokument aus der URL, und führt die gewünschte Operation aus.

© 2007 Altova GmbH

15.3.9 Aktive Datei hinzufügen

Mit diesem Befehl fügen Sie die aktive Datei in das aktuelle Projekt ein. Wenn Sie eine Datei von der Festplatte oder über eine URL geöffnet haben, können Sie die Datei mit diesem Befehl in das aktuelle Projekt aufnehmen.

15.3.10 Aktive und verwandte Dateien hinzufügen

Mit diesem Befehl fügen Sie das aktive XML-Dokument und alle verwandten Dateien in das Projekt ein. Beim Arbeiten mit einem XML-Dokument, das auf einem Schema oder einer DTD basiert, fügt dieser Befehl nicht nur das XML-Dokument zum Projekt hinzu sondern auch alle verwandten Dateien (z.B. DTDs und sämtliche geparste Entitäten, auf die die DTD referenziert).

Bitte beachten Sie: Dateien, die durch Processing Instructions referenziert werden (z.B. XSLT-Dateien) werden nicht als verwandte Dateien behandelt.

15.3.11 Projektordner in Projekt einfügen...

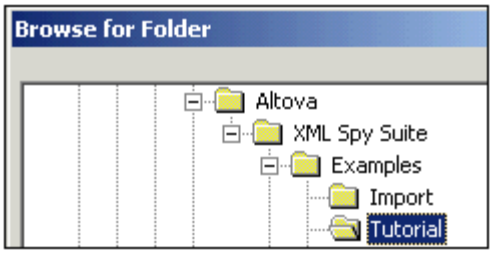
Mit diesem Befehl fügen Sie einen neuen Ordner in das aktuelle Projekt ein. Verwenden Sie diesen Befehl, um einen neuen Ordner oder Unterordner im aktuellen Projekt anzulegen. Sie können diesen Befehl auch über das Kontextmenü (Rechtsklick auf einen Ordner im Projektfenster) aufrufen.

15.3.12 Externes Verzeichnis zum Projekt hinzufügen...

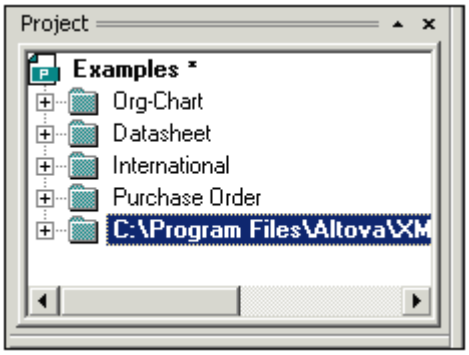
Mit diesem Befehl fügen Sie einen neues **externes Verzeichnis** in das aktuelle Projekt ein. Verwenden Sie diese Funktion, um ein lokales Verzeichnis (oder ein Netzwerkverzeichnis) in das aktuelle Projekt einzufügen. Sie können diesen Befehl auch über das Kontextmenü aufrufen (Rechtsklick auf ein Verzeichnis im Projektfenster). Bitte beachten Sie, dass Dateien in externen Verzeichnissen nicht unter Versionskontrolle gestellt werden können.

So fügen Sie ein externes Verzeichnis in ein Projekt ein:

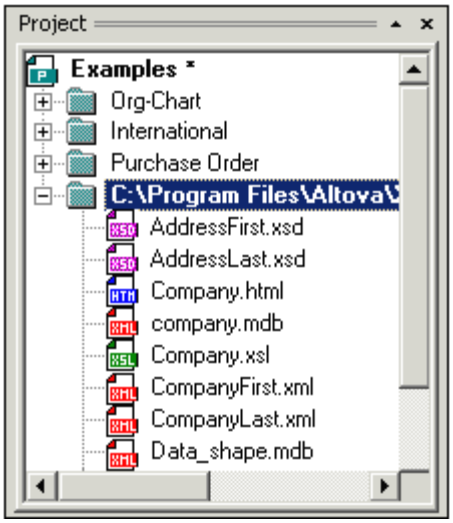
1. Wählen Sie die Menüoption **Projekt | Externes Verzeichnis zu Projekt hinzufügen...**
2. Wählen Sie das Verzeichnis, welches hinzugefügt werden soll (Dialogfeld "Ordner suchen"), und klicken Sie auf OK zum Bestätigen.



Das ausgewählte Verzeichnis wird nun im Projektfenster angezeigt.

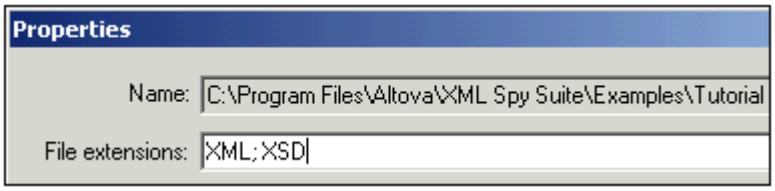


3. Klicken Sie auf das Symbol "+", um den Inhalt des Verzeichnisses anzuzeigen.



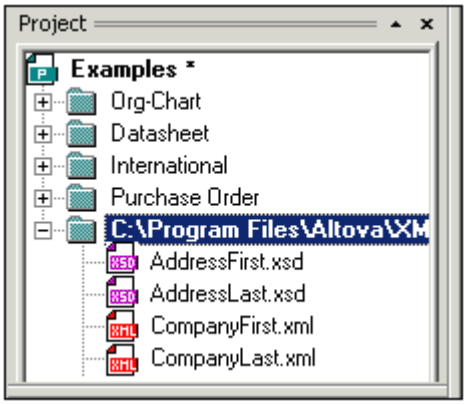
So filtern Sie den Inhalt eines Verzeichnisses:

1. Rechtsklicken Sie auf den lokalen Ordner. Wählen Sie im Popup-Menü die Option **Eigenschaften**.
   Das Dialogfeld Eigenschaften wird geöffnet.



2. Klicken Sie auf das Feld **Erweiterungen**, und geben Sie die Erweiterungen der Dateien an, die angezeigt werden sollen. Bei mehreren Erweiterungen trennen Sie diese durch ein **Semikolon** (in diesem Beispiel XML und XSD Schema).
3. Klicken Sie auf OK um Ihre Auswahl zu bestätigen.

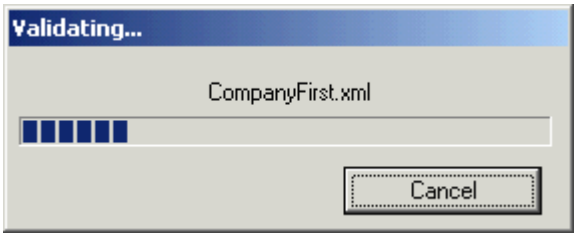
© 2007 Altova GmbH



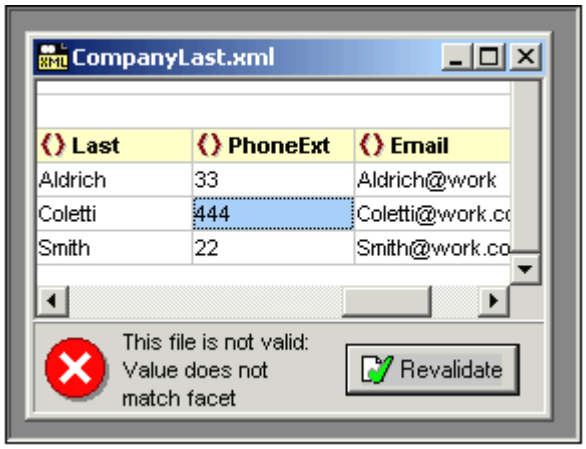
Im Projektfenster werden jetzt ausschließlich XML und XSD Dateien im Verzeichnis "Tutorial" angezeigt.

So prüfen Sie ein externes Verzeichnis auf Gültigkeit und Wohlgeformtheit:
So validieren Sie ein externes Verzeichnis und prüfen es auf Wohlgeformtheit:

1. Wählen Sie aus, welche Dateiarten angezeigt werden sollen oder markieren Sie diese im externen Ordner.
2. Klicken Sie auf das Verzeichnis und anschließend auf das Symbol "Wohlgeformtheit prüfen" oder "Validieren" (Kürzel F7, bzw. F8)
   Alle im Ordner angezeigten Dateien werden nun überprüft.



Falls eine Datei nicht wohlgeformt oder ungültig ist, wird diese im Hauptfenster geöffnet. Hier kann der Fehler korrigiert werden. In diesem Beispiel wird die Datei CompanyLast geöffnet, da der Wert PhoneExt nicht mit dem definierten Facet, der im zugrundeliegenden Schema definiert wurde, übereinstimmt. (Das Schema erlaubt nur zweistellige Ziffern für diesen Wert.)



3. Korrigieren Sie den Fehler, und starten Sie die Überprüfung erneut, um das restliche Verzeichnis zu kontrollieren.

Bitte beachten Sie: Sie können mehrere Dateien gleichzeitig im Verzeichnis markieren, wenn Sie die Strg-Taste gedrückt halten, und die gewünschten Dateien anklicken. In diesem Fall werden beim Betätigen der Tasten F7 oder F8 nur die ausgewählten Dateien geprüft.

So aktualisieren Sie den Inhalt des Projektverzeichnisses:
In lokalen oder netzwerkgebundenen Verzeichnissen können jederzeit Dateien gelöscht oder hinzugefügt werden. Um die Verzeichnis-Ansicht zu aktualisieren,

- Rechtsklicken Sie auf das externe Verzeichnis, und wählen Sie im Popup-Menü die Option **Externes Verzeichnis aktualisieren**.

Dateien oder Verzeichnisse löschen:

- Um ein Verzeichnis aus dem Projektfenster zu löschen, klicken Sie auf das **Verzeichnis** und drücken Sie die **Entf**-Taste. Das Verzeichnis wird nur aus der Projekt-Ansicht entfernt; auf der Festplatte oder im Netzwerk ist das Verzeichnis weiterhin vorhanden.
- Durch Klick auf eine **einzelne Datei** und **Drücken der Entf-Taste** wird die Datei **nicht aus dem Projektfenster gelöscht**. Sie müssen sie physisch löschen und dann den Inhalt des externen Ordners aktualisieren.

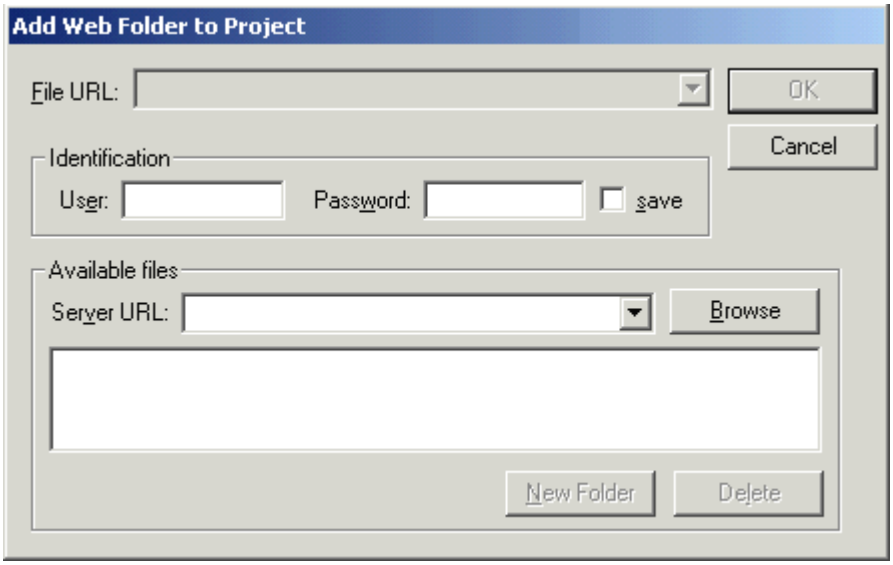
15.3.13 Externes Webverzeichnis zum Projekt hinzufügen...

Mit diesem Befehl fügen Sie ein **externes Webverzeichnis** in das aktuelle Projekt ein.

Verwenden Sie diese Funktion, um ein neues Webverzeichnis in das aktuelle Projekt einzufügen. Sie können diesen Befehl auch über das Kontextmenü (Rechtsklick auf ein Verzeichnis im Projektfenster) aufrufen. Bitte beachten Sie, dass Dateien in externen Verzeichnissen nicht unter Versionskontrolle gestellt werden können.

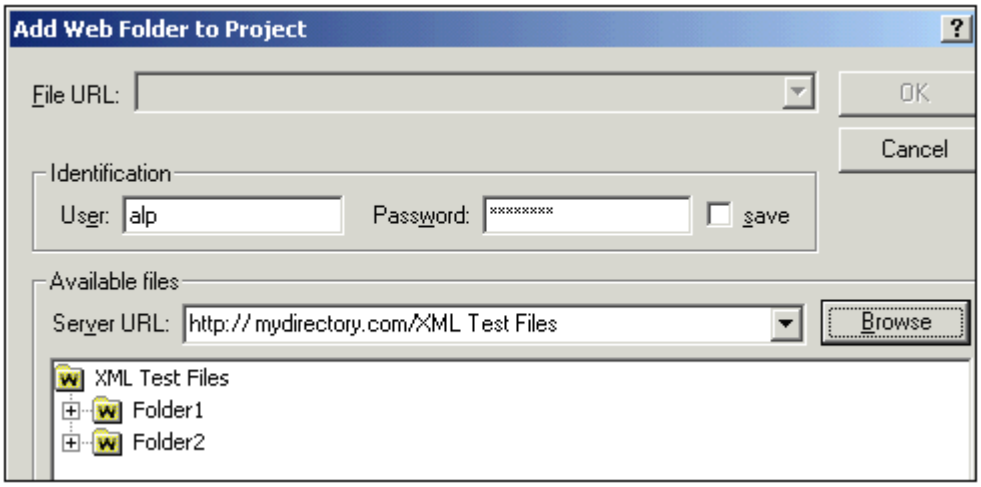
So fügen Sie ein externes Webverzeichnis in ein Projekt ein:

1. Wählen Sie die Menüoption **Projekt | Externes Webverzeichnis zum Projekt hinzufügen**.
   Das Dialogfeld "Webordner in Projekt einfügen" wird angezeigt.

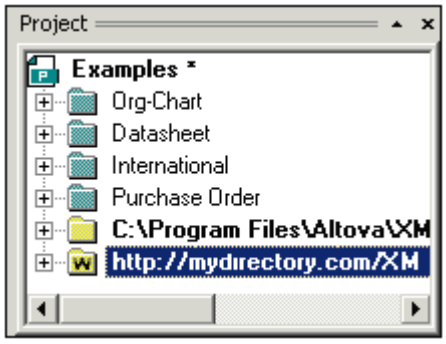


2. Klicken Sie auf das Server URL-Feld, um die URL anzugeben, und geben Sie Benutzernamen und Passwort in die entsprechenden Felder ein.
3. Klicken Sie auf "Durchsuchen" um die Verbindung mit dem Server herzustellen, und die verfügbaren Dateien zu anzuzeigen.

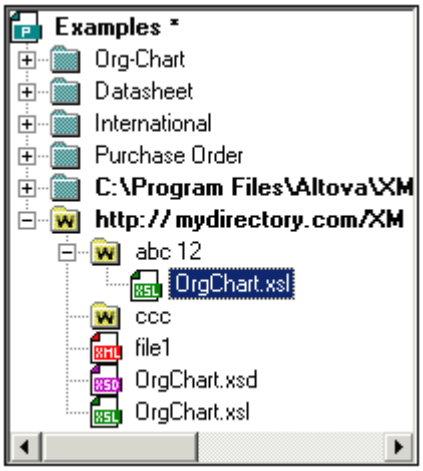
 © 2007 Altova GmbH



4. Klicken Sie auf das Verzeichnis, welches Sie in die Projektansicht übernehmen möchten. Die OK Schaltfläche wird erst danach aktiv. Verzeichnisname und http: Serveradresse werden nun im Feld Datei-URL angezeigt.
5. Klicken Sie auf **OK**, um das Verzeichnis in das Projekt zu übernehmen.



6. Klicken Sie auf das Symbol "+" um den Inhalt des Verzeichnisses anzuzeigen.



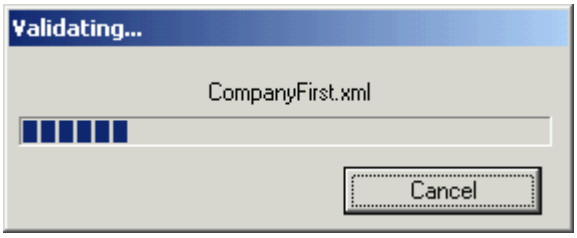
So filtern Sie den Inhalt eines Verzeichnisses:

1. **Rechtsklicken** Sie auf das Webverzeichnis, und wählen Sie im Popup-Menü die Option **Eigenschaften**.
   Das Dialogfeld Eigenschaften wird geöffnet.

2. Klicken Sie auf das Feld **Erweiterungen**, und geben Sie die Erweiterungen der Dateien an, die angezeigt werden sollen. Bei mehreren Erweiterungen trennen Sie diese durch ein **Semikolon** (z.B. XML und Schema XSDs).
3. Klicken Sie auf OK um Ihre Auswahl zu bestätigen.
   Im Projektfenster werden jetzt ausschließlich die XML und XSD Dateien des Webverzeichnisses angezeigt.

So prüfen Sie ein Verzeichnis auf Gültigkeit und Wohlgeformtheit:

1. Klicken Sie auf das Verzeichnis und anschließend auf das Symbol "Wohlgeformtheit prüfen" oder "Validieren" (Kürzel F7, bzw. F8)
   Alle angezeigten Dateien im Verzeichnis werden nun überprüft.



   Falls eine Datei nicht wohlgeformt oder ungültig ist, wird diese im Hauptfenster geöffnet. Hier können Sie den Fehler korrigieren.
2. Nach erfolgter Korrektur, starten Sie die Prüfung erneut, um das restliche Verzeichnis zu kontrollieren.

Bitte beachten Sie:
Sie können mehrere Dateien gleichzeitig im Verzeichnis markieren, wenn Sie die Strg-Taste gedrückt halten, und die gewünschten Dateien anklicken. In diesem Fall werden beim Betätigen der Funktionstasten F7 oder F8 nur die markierten Dateien überprüft.

So aktualisieren Sie den Inhalt des Projektverzeichnisses:
Sie können jederzeit Dateien aus dem Webverzeichnis hinzufügen oder löschen. Um die Verzeichnis-Ansicht zu aktualisieren,

- Rechtsklicken Sie auf das externe Verzeichnis, und wählen Sie im Popup-Menü die Option **Externes Verzeichnis aktualisieren**.

Dateien oder Verzeichnisse löschen:

- Um ein Verzeichnis aus dem Projektfenster zu löschen, klicken Sie auf das **Verzeichnis** und drücken Sie die **Entf**-Taste. Das Verzeichnis wird nur aus der Projekt-Ansicht entfernt; auf der Festplatte oder im Netzwerk ist das Verzeichnis weiterhin vorhanden.
- Durch Klick auf eine **einzelne Datei** und Betätigen der **Entf**-Taste wird die Datei **nicht aus dem Projektfenster gelöscht**. Sie müssen sie physisch löschen, und dann den Inhalt des externen Ordners aktualisieren.

15.3.14 Projekteigenschaften...

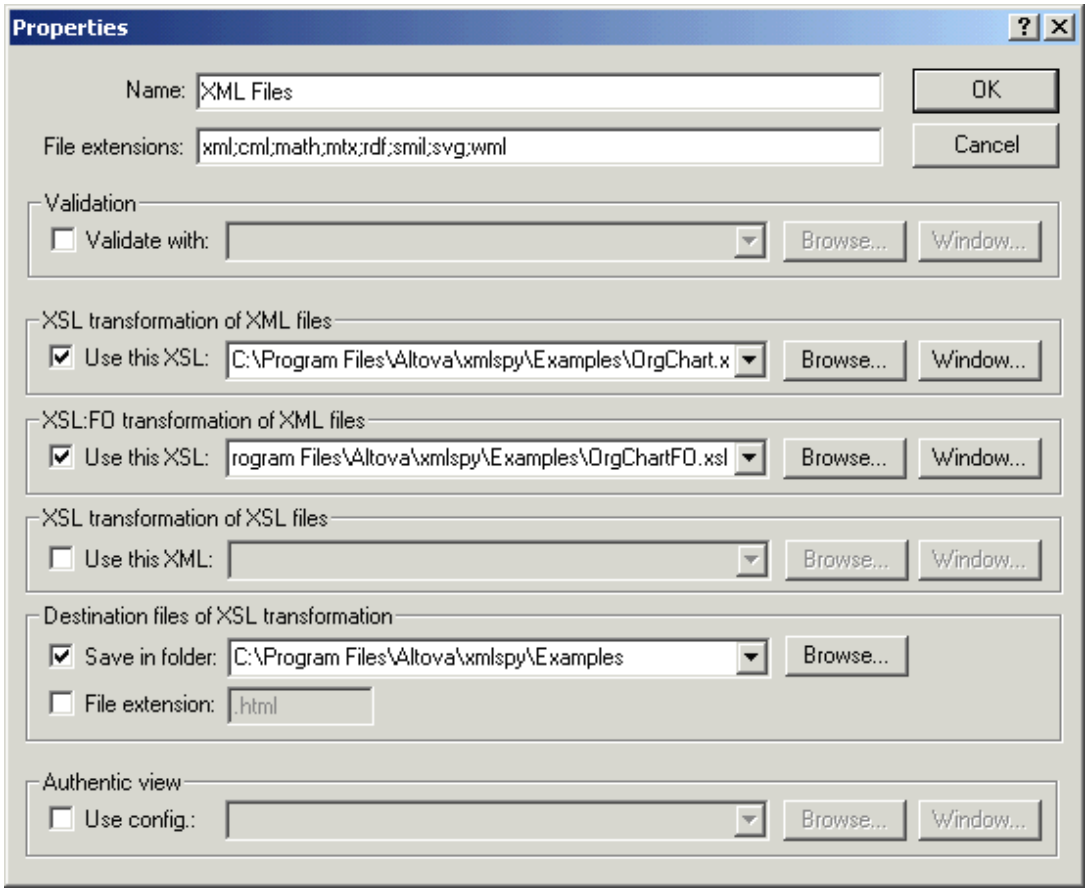
Mit diesem Befehl können Sie wichtige Einstellungen für spezifische Ordner Ihres Projekts definieren.

So definieren Sie Projekteigenschaften für ein Verzeichnis:

1. Rechtsklicken Sie auf das Verzeichnis, für welches Sie Einstellungen vornehmen

möchten, und
2. Wählen Sie den Befehl **Eigenschaften...** aus dem Kontextmenü.

Bitte beachten Sie:
Falls die Projektdatei unter Versionskontrolle steht, werden Sie gefragt, ob die Projektdatei (*.spp) ausgecheckt werden soll. Nach dem Ändern der Einstellungen klicken Sie auf OK um diese zu speichern.



Die Zuweisungen in den Feldern "XSL-Vorlage" bzw. "XML-Vorlage" haben Vorrang vor anderen lokalen Zuweisungen innerhalb einer XML-Datei. So wird z.B. die Datei OrgChart.xsl ("XSL-Vorlage") immer dann verwendet, wenn XML-Dateien im **XML-Dateiverzeichnis** transformiert werden. Die hier vorgenommene Zuweisung hat auch Vorrang vor anderen Zuweisungen, die für übergeordnete Ordner vorgenommen wurden.

Dateierweiterungen
Die Dateierweiterungen helfen beim Hinzufügen neuer Dateien zum Projekt (im Gegensatz zum Einordnen in einen bestimmten Ordner) bei der automatischen Einordnung der Dateien in Ordner eines Projekts.

Validierung
Definieren Sie das DTD- oder Schemadokument, welches bei der Validierung aller Dateien des aktuellen Verzeichnisses verwendet werden soll (in diesem Beispiel die Hauptseiten).

XSL Transformation von XML-Dateien
Definieren Sie das XSL Stylesheet, welches zur XSL Transformation aller Dateien eines Verzeichnisses verwendet werden soll.

Wenn Sie selbst XSL Stylesheets erstellen, können Sie ein XML-Beispieldokument zuweisen, das als Vorschau für das XSL Stylesheet bei der XML-Transformation dient. Sie können dies anstelle des XML Instanzdokuments verwenden.

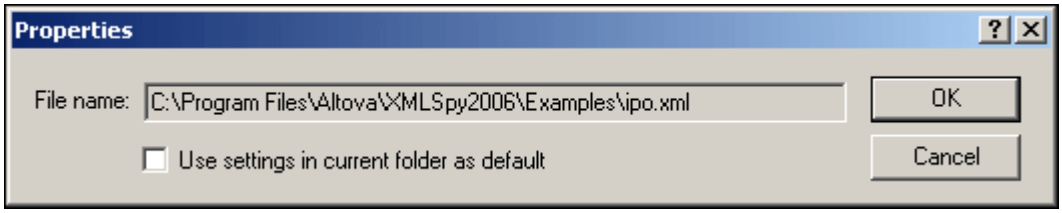
XSL:FO Transformation von XML-Dateien
Sie können das XSL Stylesheet mit XSL:FO Markup definieren, das für eine XSL:FO Transformation aller Dateien eines Verzeichnisses verwendet werden soll.

Zieldateien bei XSL Transformation
Für Batchtransformationen in XSL können Sie das Zielverzeichnis angeben, in dem die transformierten Dateien abgelegt werden sollen.

Wenn Sie eine Datei oder URL in mehreren Verzeichnissen Ihres Projekts abgelegt haben, können Sie über das Dialogfeld "Eigenschaften" das Standardverzeichnis definieren, mit dessen Einstellungen die Validierung oder Transformation im non-batch Modus erfolgen soll. Aktivieren Sie dazu das Kontrollkästchen **Einstellungen des aktuellen Ordners als Standard verwenden** (siehe Screenshot).

So rufen Sie das Dialogfeld "Eigenschaften" auf und aktivieren das Kontrollkästchen:

1. Kopieren Sie eine XML-Datei in einem Projekt in einen anderen Ordner.
2. Rechtsklicken Sie im Projektfenster auf die kopierte Datei und wählen Sie im Kontextmenü den Befehl **Eigenschaften.**



Authentic
Die Option "Konfig. verwenden" ermöglicht es Ihnen, das StyleVision Power Stylesheet (SPS Datei) im aktuellen Verzeichnis auszuwählen, welche beim Editieren von XML Dateien in der Authentic-Ansicht zugrunde gelegt werden soll. Nachdem Sie die Schema-, SPS- und XML-Dateien miteinander verknüpft und in ein Projekt eingegeben haben, kann ein Verschieben dieser Dateien in einen anderen Ordner Fehler bei den Verknüpfungen verursachen.

Um solche Fehler zu vermeiden, sollten Sie am besten einen endgültigen Speicherort für Ihre Schema-, SPS- und XML-Dateien festlegen, bevor Sie sie miteinander verknüpfen und einem Projekt zuweisen. Weitere Informationen zur Verwendung der Authentic-Ansicht finden Sie im Handbuch zur Authentic-Ansicht.

15.3.15 Zuletzt verwendete Projekte

Mit dieser Funktion werden die neun zuletzt verwendeten Projekte angezeigt um einen schnelleren Zugriff auf diese zu erhalten.

Beachten Sie bitte, dass XMLSpy automatisch das zuletzt verwendete Projekt beim Starten von XMLSpy öffnen kann. (**Extras | Optionen|** Register **Datei**, Projekt | Letztes Projekt beim Programmstart öffnen).

© 2007 Altova GmbH

15.4 Menü "XML"

Das XML-Menü enthält alle nötigen Befehle um mit XML-Dokumenten zu arbeiten. Sie können Elemente einfügen oder anhängen, Elementhierarchien ändern, ein Namespace Präfix für Objekte setzen und XPaths im Zusammenhang mit einzelnen XML-Dokumenten evaluieren.

Zu den am häufigsten verwendeten XML-Funktionen zählen die Überprüfung von Dokumenten auf ihre Wohlgeformtheit und Gültigkeit. Sie finden diese Befehle in diesem Menü.

15.4.1 Einfügen

Der Befehl **Einfügen** ist zwar in allen Menüs aktiv, kann aber nur in der Grid-Ansicht verwendet werden. Er enthält ein Untermenü (*siehe Screenshot*), über das Sie folgende Möglichkeiten haben:

- die XML-Deklaration und Node-Typen (Attribut, Element, Text, CDATA, Comment, Processing Instruction) in XML-Dokumente einfügen,
- DOCTYPE-Deklarationen und externe DTD-Deklarationen in XML-Dokumente einfügen und
- DTD-Deklarationen (ELEMENT, ATTLIST, ENTITY und NOTATION) in DTD-Dokumente und interne DTD-Deklarationen von XML-Dokumenten einfügen.

Attribut einfügen

Strg+Umschalt+I

Diese Funktion ist nur in der Grid-Ansicht aktiv und fügt ein neues **Attribut** vor dem ausgewählten Datenelement ein. Ein eingefügtes Attribut kann in der Grid-Ansicht einige Zeilen vor dem aktuellen Datenelement angezeigt werden, da Attribute in der Grid-Ansicht unmittelbar nach deren Parent-Element und vor allen Child-Elementen dieses Parent-Elements angezeigt werden.

Element einfügen

Strg+Umschalt+E

Diese Funktion steht nur in der Grid-Ansicht zur Verfügung und fügt ein neues **Element** vor dem ausgewählten Datenelement ein. Falls die Auswahl ein Attribut ist, wird das neue Element vor dem ersten Child-Element des Parent-Elements des Attributs eingefügt.

Text einfügen

Strg+Umschalt+T

Diese Funktion steht nur in der Grid-Ansicht zur Verfügung und fügt eine neue Textzeile vor dem ausgewählten Datenelement ein. Falls die Auswahl ein Attribut ist, wird die Textzeile nach dem Attribut und vor dem ersten Child-Element des Parent-Elements des Attributs eingefügt.

CDATA einfügen

Strg+Umschalt+D

Diese Funktion steht nur in der Grid-Ansicht zur Verfügung und fügt einen neuen CDATA-Block vor dem ausgewählten Datenelement ein. Falls die Auswahl ein Attribut ist, wird der CDATA-Block nach dem Attribut und vor dem ersten Child-Element des Parent-Elements des Attributs eingefügt.

Comment einfügen

Strg+Umschalt+M

Diese Funktion steht nur in der Grid-Ansicht zur Verfügung und fügt einen neuen **Comment** vor dem ausgewählten Datenelement ein. Falls die Auswahl ein Attribut ist, wird die neue Comment-Zeile nach dem Attribut und vor dem ersten Child-Element des Parent-Elements des Attributs eingefügt.

XML einfügen

Diese Funktion steht nur in der Grid-Ansicht zur Verfügung und fügt eine Zeile für die XML-Deklaration vor dem ausgewählten Datenelement ein. Sie müssen die Child-Attribute der XML-Deklaration und die Werte dieser Attribute einfügen. Eine XML-Deklaration muss ungefähr folgendermaßen aussehen:

```
<?xml version="1.0" encoding="UTF-8"?>
```

Bitte beachten Sie: Da ein XML-Dokument nur eine XML-Deklaration am Anfang der Datei enthalten darf, sollten Sie diesen Befehl nur verwenden, wenn die erste Zeile des Dokuments ausgewählt ist und wenn das Dokument noch keine XML-Deklaration enthält.

Processing Instruction einfügen

Diese Funktion steht nur in der Grid-Ansicht zur Verfügung und fügt eine neue Processing Instruction (PI) vor dem ausgewählten Datenelement ein. Falls die Auswahl ein Attribut ist, wird die PI nach dem Attribut und vor dem ersten Child-Element des Parent-Elements des Attributs eingefügt.

DOCTYPE einfügen

Der Befehl **DOCTYPE einfügen** steht in der Grid-Ansicht einer XML-Datei zur Verfügung, wenn der Node der obersten Ebene ausgewählt ist. Sie hängen damit eine DOCTYPE-Deklaration am Beginn des XML-Dokuments an. Sie müssen den Namen des DOCTYPE angeben und dieser Name muss mit dem Namen des Dokument-Elements übereinstimmen.

© 2007 Altova GmbH

Nach Eingabe des DOCTYPE-Namens können Sie die Deklarationen eingeben, die in der internen DTD-Untermenge verwendet werden sollen.

Bitte beachten Sie:

- Die DOCTYPE-Deklaration darf nur zwischen der XML-Deklaration und dem XML-Dokument-Element stehen.
- Sie können stattdessen auch den Befehl DTD zuweisen verwenden, mit dem Sie eine DOCTYPE-Anweisung erstellen, die auf ein externes DTD-Dokument referenziert.

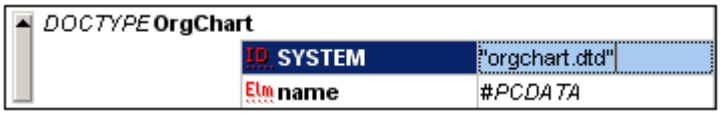
ExternalID einfügen

Eine DOCTYPE-Deklaration in einer XML-Datei kann eine Referenz auf eine externe Ressource, die eine DTD-Deklaration enthält, beinhalten. Diese Ressource wird entweder über einen Public- oder einen System Identifier referenziert.
Beispiel:

```
<! DOCTYPE doc_element_name PUBLIC "publicID" "systemID">
<! DOCTYPE doc_element_name SYSTEM "systemID">
```

Ein System Identifier ist eine URI, die eine externe Ressource spezifiziert. Der Public Identifier ist ein pfadunabhängiger Identifier, der dazu dient, die Referenz auf eine externe Ressource aufzuheben. So sind z.B. in Ihrem XMLSpy Installationsverzeichnis URIs für häufig verwendete DTDs und XML-Schemas in einer Katalogdatei mit dem Namen `MainCatalog.xml` gespeichert. Um die Referenz auf eine in `MainCatalog.xml` aufgelistete DTD aufzuheben, können Sie einen Public Identifier verwenden.

Der Befehl **ExternalID einfügen** steht zur Verfügung, wenn in der Grid-Ansicht in einer XML-Datei ein Child-Datenelement der DOCTYPE-Deklaration ausgewählt ist. Mit diesem Befehl fügen Sie in der Grid-Ansicht eine Zeile für einen externen Identifier (`PUBLIC` oder `SYSTEM`) ein. Sie müssen die Art des Identifiers und seinen Wert angeben.



In der Textansicht, die der oben gezeigten Abbildung aus der Grid-Ansicht entspricht, sieht dies aus wie folgt:

```
<! DOCTYPE OrgChart SYSTEM "orgchart.dtd" [
  <! ELEMENT name (#PCDATA)>
]>
```

Bitte beachten Sie: Sie können eine Zeile für eine External ID als Child hinzufügen, wenn das Datenelement DOCTYPE ausgewählt ist oder Sie können diese einfügen oder anhängen, wenn eines der Child-Elemente des Datenelements DOCTYPE, z.B. die ELEMENT-Deklaration `name` im obigen Beispiel, ausgewählt ist.

ELEMENT einfügen

Diese Funktion steht nur in der Grid-Ansicht und nur für DTD-Dokumente zur Verfügung oder, wenn in der DOCTYPE-Deklaration eines XML-Dokuments ein Datenelement ausgewählt ist. Der Befehl fügt eine ELEMENT-Deklaration vor der ausgewählten Deklaration ein.

ATTLIST einfügen

Diese Funktion steht nur in der Grid-Ansicht und nur für DTD-Dokumente zur Verfügung oder, wenn in der DOCTYPE-Deklaration eines XML-Dokuments ein Datenelement ausgewählt ist. Der Befehl fügt eine ATTLIST-Deklaration vor der ausgewählten Deklaration ein.

ENTITY einfügen

Diese Funktion steht nur in der Grid-Ansicht und nur für DTD-Dokumente zur Verfügung oder, wenn in der DOCTYPE-Deklaration eines XML-Dokuments ein Datenelement ausgewählt ist. Der Befehl fügt eine ENTITY-Deklaration vor der ausgewählten Deklaration ein.

NOTATION einfügen

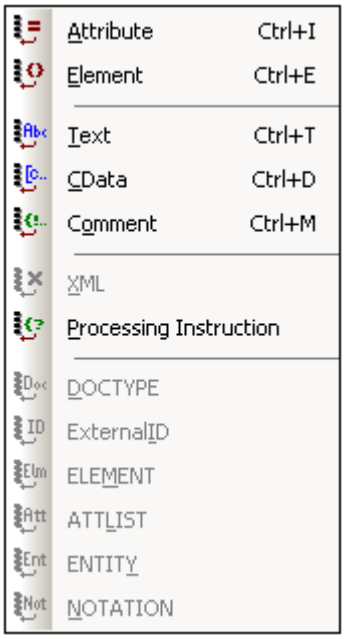
Diese Funktion steht nur in der Grid-Ansicht und nur für DTD-Dokumente zur Verfügung oder, wenn in der DOCTYPE-Deklaration eines XML-Dokuments ein Datenelement ausgewählt ist. Der Befehl fügt eine NOTATION-Deklaration vor der ausgewählten Deklaration ein.

15.4.2 Anhängen

Der Befehl **Anhängen** ist zwar in allen Menüs aktiv, kann aber nur in der Grid-Ansicht verwendet werden. Er enthält ein Untermenü (*siehe Screenshot*), über das Sie folgende Möglichkeiten haben:

- die XML-Deklaration und Node-Typen (Attribut, Element, Text, CDATA, Comment, Processing Instruction) in XML-Dokumenten anhängen,
- DOCTYPE-Deklarationen und externe DTD-Deklarationen in XML-Dokumenten anhängen und
- DTD-Deklarationen (ELEMENT, ATTLIST, ENTITY und NOTATION) in DTD-Dokumenten und internen DTD-Deklarationen von XML-Dokumenten anhängen.

© 2007 Altova GmbH



Attribut anhängen

Strg+I

Diese Funktion steht nur in der Grid-Ansicht zur Verfügung und hängt ein neues Attribut an.

Element anhängen

Strg+E

Diese Funktion steht nur in der Grid-Ansicht zur Verfügung und hängt ein neues **Element** nach dem letzten Geschwisterelement des ausgewählten Elements an. Falls es sich bei der aktuellen Auswahl um einen Attribut-Node handelt, wird der Element-Node nach dem letzten Child des Parent-Elements des Attributs angehängt.

Text anhängen

Strg+T

Diese Funktion steht nur in der Grid-Ansicht zur Verfügung und hängt einen neuen Textblock nach dem letzten Geschwisterelement des ausgewählten Elements an. Falls es sich bei der Auswahl um einen Attribut-Node handelt, wird der Textblock nach letzten Child des Parent-Elements des Attributs angehängt.

CDATA anhängen

Strg+D

Diese Funktion steht nur in der Grid-Ansicht zur Verfügung und hängt einen CDATA Node nach dem letzten Geschwisterelement eines ausgewählten Node (mit Ausnahme eines Attribut-Node) an. Falls es sich bei der Auswahl um einen Attribut-Node handelt, wird der CDATA-Abschnitt nach letzten Child des Parent-Elements des ausgewählten Attributs angehängt.

Comment anhängen

Strg+M

Diese Funktion steht nur in der Grid-Ansicht zur Verfügung und hängt einen Comment Node nach dem letzten Geschwisterelement eines ausgewählten Node (mit Ausnahme eines Attribut-Node) an. Falls es sich bei der Auswahl um einen Attribut-Node handelt, wird der Comment Node nach letzten Child des Parent-Elements des ausgewählten Attributs angehängt.

XML anhängen

Mit dieser Funktion fügen Sie eine neue XML-Deklaration `<?xml version="1.0" encoding="UTF-8"?>` als das erste Datenelement in einem Dokument ein. Bitte beachten Sie, dass ein XML-Dokument nur eine XML-Deklaration enthalten darf, welche am Beginn der Datei stehen muss.

Processing Instruction anhängen

Diese Funktion steht nur in der Grid-Ansicht zur Verfügung und hängt einen Processing Instruction Node nach dem letzten Geschwisterelement eines ausgewählten Node (mit Ausnahme eines Attribut-Node) an. Falls es sich bei der Auswahl um einen Attribut-Node handelt, wird der Processing Instruction Node nach letzten Child des Parent-Elements des ausgewählten Attributs angehängt.

DOCTYPE anhängen

Der Befehl **DOCTYPE anhängen** steht in der Grid-Ansicht einer XML-Datei zur Verfügung, wenn der Node der obersten Ebene ausgewählt ist. Sie hängen damit eine DOCTYPE-Deklaration am Beginn des XML-Dokuments an. Sie müssen den Namen des DOCTYPE angeben und dieser Name muss mit dem Namen des Dokument-Elements übereinstimmen.

© 2007 Altova GmbH

Nach Eingabe des DOCTYPE-Namens können Sie die Deklarationen eingeben, die in der internen DTD-Untermenge verwendet werden sollen.

Bitte beachten Sie:

- Die DOCTYPE-Deklaration darf nur zwischen der XML-Deklaration und dem XML-Dokument-Element stehen.
- Sie können stattdessen auch den Befehl DTD zuweisen verwenden, mit dem Sie eine DOCTYPE-Anweisung erstellen, die auf ein externes DTD-Dokument referenziert.

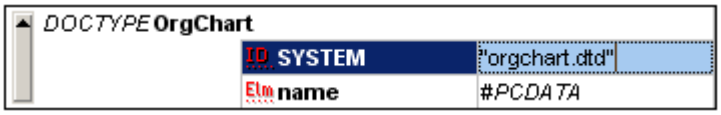
ExternalID anhängen

Eine DOCTYPE-Deklaration in einer XML-Datei kann eine Referenz auf eine externe Ressource, die eine DTD-Deklaration enthält, beinhalten. Diese Ressource wird entweder über einen Public- oder eine System Identifier referenziert.
Beispiel:

```
<!DOCTYPE doc_element_name PUBLIC "publicID" "systemID">
<!DOCTYPE doc_element_name SYSTEM "systemID">
```

Ein System Identifier ist eine URI, die eine externe Ressource spezifiziert. Der Public Identifier ist ein pfadunabhängiger Identifier, der dazu dient, die Referenz auf eine externe Ressource aufzuheben. So sind z.B. in Ihrem XMLSpy Installationsverzeichnis URIs für häufig verwendete DTDs und XML-Schemas in einer Katalogdatei mit dem Namen `MainCatalog.xml` gespeichert. Um die Referenz auf eine in `MainCatalog.xml` aufgelistete DTD aufzuheben, können Sie einen Public Identifier verwenden.

Der Befehl **ExternalID anhängen** steht zur Verfügung, wenn in der Grid-Ansicht in einer XML-Datei ein Child-Datenelement der DOCTYPE-Deklaration ausgewählt ist. Mit diesem Befehl fügen Sie in der Grid-Ansicht eine Zeile für einen externen Identifier (`PUBLIC` oder `SYSTEM`) ein. Sie müssen die Art des Identifiers und seinen Wert angeben.



In der Textansicht, die der oben gezeigten Abbildung aus der Grid-Ansicht entspricht, sieht dies aus wie folgt:

```
<!DOCTYPE OrgChart SYSTEM "orgchart.dtd" [
 <!ELEMENT name (#PCDATA)>
]>
```

Bitte beachten Sie: Sie können eine Zeile für eine External-ID als Child hinzufügen, wenn das Datenelement DOCTYPE ausgewählt ist oder Sie können diese einfügen oder anhängen, wenn eines der Child-Elemente des Datenelements DOCTYPE, z.B. die ELEMENT-Deklaration `name` im obigen Beispiel, ausgewählt ist.

ELEMENT anhängen

Diese Funktion steht nur in der Grid-Ansicht und nur für DTD-Dokumente zur Verfügung oder, wenn in der DOCTYPE-Deklaration eines XML-Dokuments ein Datenelement ausgewählt ist. Der Befehl hängt eine ELEMENT-Deklaration an die Liste der Deklarationen an.

ATTLIST anhängen

Diese Funktion steht nur in der Grid-Ansicht und nur für DTD-Dokumente zur Verfügung oder, wenn in der DOCTYPE-Deklaration eines XML-Dokuments ein Datenelement ausgewählt ist. Der Befehl hängt eine ATTLIST-Deklaration an die Liste der Deklarationen an.

ENTITY anhängen

Diese Funktion steht nur in der Grid-Ansicht und nur für DTD-Dokumente zur Verfügung oder, wenn in der DOCTYPE-Deklaration eines XML-Dokuments ein Datenelement ausgewählt ist. Der Befehl hängt eine ENTITY-Deklaration an die Liste der Deklarationen an.

NOTATION anhängen

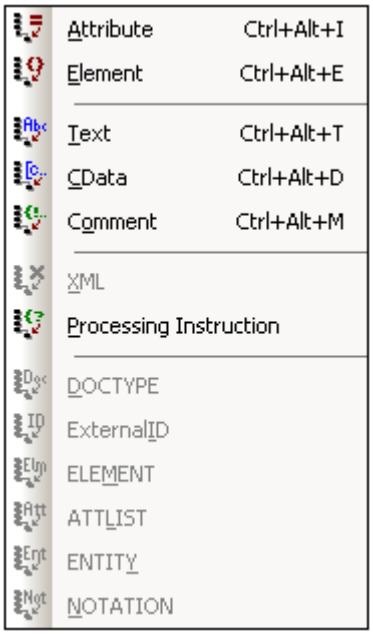
Diese Funktion steht nur in der Grid-Ansicht und nur für DTD-Dokumente zur Verfügung oder, wenn in der DOCTYPE-Deklaration eines XML-Dokuments ein Datenelement ausgewählt ist. Der Befehl hängt eine NOTATION-Deklaration an die Liste der Deklarationen an.

15.4.3 Child hinzufügen

Der Befehl **Child hinzufügen** ist zwar in allen Menüs aktiv, kann aber nur in der Grid-Ansicht verwendet werden. Der Befehl enthält ein Untermenü (*siehe Screenshot*), über das Sie die folgenden Child-Datenelemente zum aktuell ausgewählten Element hinzufügen können:

- die XML-Deklaration und Node-Typen (Attribut, Element, Text, CDATA, Comment, Processing Instruction) in XML-Dokumenten,
- DOCTYPE-Deklarationen und externe DTD-Deklarationen in XML-Dokumenten und
- DTD-Deklarationen (ELEMENT, ATTLIST, ENTITY und NOTATION) in DTD-Dokumenten und internen DTD-Deklarationen von XML-Dokumenten.

© 2007 Altova GmbH



Child-Attribut hinzufügen

Strg+ALT+I

Dieser Befehl steht nur in der Grid-Ansicht und nur bei Auswahl eines Element-Node zur Verfügung. Er fügt ein **neues Attribut** als Child des ausgewählten Element-Node ein.

Child-Element hinzufügen

Strg+ALT+E

Dieser Befehl steht nur in der Grid-Ansicht zur Verfügung. Er fügt ein neues Element als Child des ausgewählten Node ein.

Child-Text hinzufügen

Strg+ALT+T

Dieser Befehl steht nur in der Grid-Ansicht zur Verfügung. Er fügt ein neuen Textinhalt als Child des ausgewählten Datenelements ein.

Child CDATA hinzufügen

Strg+ALT+D

Dieser Befehl steht nur in der Grid-Ansicht zur Verfügung. Er fügt einen neuen

CDATA-Abschnitt als Child des ausgewählten Datenelements ein.

Child Comment hinzufügen

Strg+ALT+M

Dieser Befehl steht nur in der Grid-Ansicht zur Verfügung. Er fügt einen neuen Comment Node als Child des ausgewählten Datenelements ein.

Child XML hinzufügen

Dieser Befehl ist nur in der Grid-Ansicht und nur, wenn die Datei **leer** ist, verfügbar. Er fügt eine neue XML-Deklaration `<?xml version="1.0" encoding="UTF-8"?>` als das erstes Datenelement in ein Dokument ein. Bitte beachten Sie, dass ein XML-Dokument nur eine XML-Deklaration enthalten darf, welche am Beginn der Datei stehen muss.

Child Processsing Instruction hinzufügen

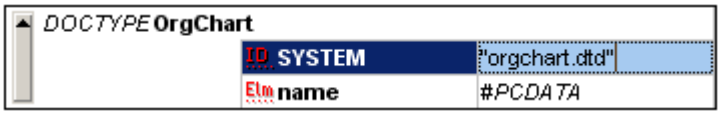
Diese Funktion steht nur in der Grid-Ansicht zur Verfügung und fügt eine neue Processing Instruction (PI) als Child des ausgewählten Datenelements ein.

Child DOCTYPE hinzufügen

Dieser Befehl steht nur in der Grid-Ansicht und nur bei einem **leeren** Dokument zur Verfügung. Er fügt eine DOCTYPE-Deklaration in ein XML-Dokument ein. Mit Hilfe der DOCTYPE-Deklaration können Sie eine interne DTD-Untermenge deklarieren.

Child ExternalID hinzufügen

Der Befehl **ExternalID hinzufügen** steht nur dann zur Verfügung, wenn in der Grid-Ansicht in einer XML-Datei die DOCTYPE-Deklaration ausgewählt ist. Mit diesem Befehl fügen Sie in der Grid-Ansicht eine Zeile für einen externen Identifier (`PUBLIC` oder `SYSTEM`) ein. Sie müssen die Art des Identifiers und seinen Wert angeben.



In der Textansicht, die der oben gezeigten Abbildung aus der Grid-Ansicht entspricht, sieht dies aus wie folgt:

```
<!DOCTYPE OrgChart SYSTEM "orgchart.dtd" [
 <!ELEMENT name (#PCDATA)>
]>
```

Bitte beachten Sie: Sie können eine Zeile für eine External-ID als Child hinzufügen, wenn das Datenelement DOCTYPE ausgewählt ist oder Sie können diese einfügen oder anhängen, wenn eines der Child-Elemente des Datenelements DOCTYPE, z.B. die ELEMENT-Deklaration `name` im obigen Beispiel, ausgewählt ist.

Child ELEMENT hinzufügen

Diese Funktion steht nur in der Grid-Ansicht und nur für DTD-Dokumente zur Verfügung oder, wenn die DOCTYPE-Deklaration eines XML-Dokuments ausgewählt ist. Der Befehl hängt eine ELEMENT-Deklaration an die Liste der Deklarationen an.

Child ATTLIST hinzufügen

Diese Funktion steht nur in der Grid-Ansicht und nur für DTD-Dokumente zur Verfügung oder, wenn die DOCTYPE-Deklaration eines XML-Dokuments ausgewählt ist. Der Befehl hängt eine ATTLIST-Deklaration an die Liste der Deklarationen an.

Child ENTITY hinzufügen

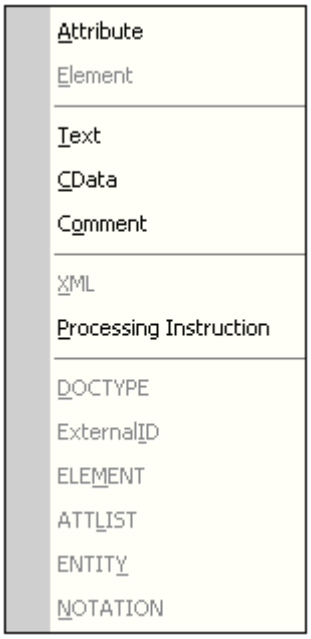
Diese Funktion steht nur in der Grid-Ansicht und nur für DTD-Dokumente zur Verfügung oder, wenn die DOCTYPE-Deklaration eines XML-Dokuments ausgewählt ist. Der Befehl hängt eine ENTITY-Deklaration an die Liste der Deklarationen an.

Child NOTATION hinzufügen

Diese Funktion steht nur in der Grid-Ansicht und nur für DTD-Dokumente zur Verfügung oder, wenn die DOCTYPE-Deklaration eines XML-Dokuments ausgewählt ist. Der Befehl hängt eine NOTATION-Deklaration an die Liste der Deklarationen an.

15.4.4 Konvertieren in

Mit diesem Befehl können Sie ein ausgewähltes Datenelement in der Grid-Ansicht in einen anderen Datenelement-Typ konvertieren. Diese Operation steht nur in der Grid-Ansicht für einzelne Datenelemente zur Verfügung, die keinen Child Node enthalten. Wenn Sie den Cursor über den Befehl **Konvertieren in** platzieren, wird ein Untermenü (*siehe Screenshot*) mit den Datenelementen angezeigt, in die das ausgewählte Datenelement konvertiert werden kann.



Bitte beachten Sie: Gehen durch die Konvertierung möglicherweise Daten verloren (z.B. geht beim Konvertieren eines Attributs in einen Comment der Attributname verloren), werden Sie mittels einer Meldung darüber informiert.

In Attribut konvertieren

Das ausgewählte Datenelement wird in ein **Attribut** konvertiert.

In Element konvertieren

Das ausgewählte Datenelement wird in ein **Element** konvertiert.

In Text konvertieren

Das ausgewählte Datenelement wird in **Text** konvertiert.

In CDATA konvertieren

Das ausgewählte Datenelement wird in einen **CDATA Block** konvertiert.

In Comment konvertieren

Das ausgewählte Datenelement wird in einen **Comment** konvertiert.

© 2007 Altova GmbH

In XML konvertieren

Mit dieser Funktion konvertieren Sie ein ausgewähltes Datenelement in eine XML Deklaration

`<?xml version="1.0" encoding="UTF-8"?>`.

Bitte beachten Sie: Jedes XML-Dokument darf nur eine XML-Deklaration enthalten, die am Beginn der Datei stehen muss.

In Processing Instruction konvertieren

Das ausgewählte Datenelement wird in eine neue **Processing Instruction** (PI) konvertiert.

In DOCTYPE konvertieren

Mit diesem Befehl konvertieren Sie das ausgewählte Datenelement in eine DOCTYPE-Deklaration (in einer XML-Datei).

Bitte beachten Sie: Die DOCTYPE-Deklaration darf nur am Beginn eines XML-Instanzdokuments zwischen der XML-Deklaration und dem Dokument-Element des XML-Dokuments stehen.

In ExternalID konvertieren

Das ausgewählte Datenelement wird in eine externe DTD-Referenz in einer DOCTYPE-Deklaration (`PUBLIC` oder `SYSTEM` Identifier) konvertiert.

In ELEMENT konvertieren

Das ausgewählte Datenelement wird in eine Element-Deklaration in einer DOCTYPE-Deklaration oder einer externen DTD konvertiert.

In ATTLIST konvertieren

Das ausgewählte Datenelement wird in eine Attributlisten-Deklaration in einer DOCTYPE Deklaration oder einer externen DTD konvertiert.

In ENTITY konvertieren

Das ausgewählte Datenelement wird in eine Entity-Deklaration in einer DOCTYPE Deklaration oder einer externen DTD konvertiert.

In NOTATION konvertieren

Das ausgewählte Datenelement wird in eine Notationsdeklaration in einer DOCTYPE Deklaration oder einer externen DTD konvertiert.

15.4.5 Tabelle

Der Befehl **Tabelle** ist zwar in allen Ansichten aktiv, kann aber nur in der Grid-Ansicht verwendet werden. Er enthält ein Untermenü mit allen für die Datenbank/Tabellen-Ansicht der Grid-Ansicht relevanten Befehle.

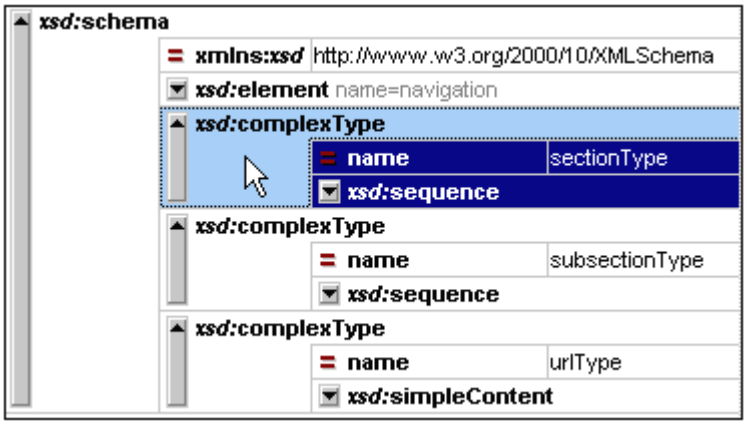
Als Tabelle anzeigen

F12

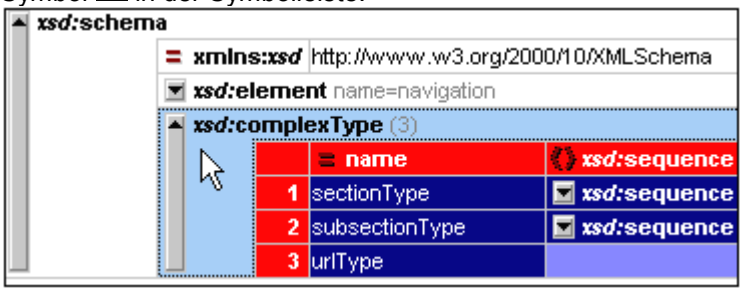
Mit diesem Befehl können Sie zwischen der Grid-Standardansicht und der Datenbank/Tabellen-Ansicht (oder Tabellenansicht) eines Dokument-Elements wechseln. In der Tabellenansicht werden sich wiederholende Elemente als Tabelle angezeigt, wobei die Zeilen die Instanzen enthalten und die Spalten die Child Nodes (einschl. Comments, CDATA-Abschnitte und PIs).

So wechseln Sie zur Tabellenansicht:

1. Markieren Sie ein sich wiederholendes Element, das in Tabellenform angezeigt werden soll.



2. Wählen Sie **XML | Als Tabelle anzeigen**, drücken Sie **F12** oder klicken Sie auf das Symbol in der Symbolleiste.



Das Element wird als Tabelle dargestellt und das Symbol ist aktiv.

Um von der Tabellenansicht eines Dokument-Elements zur normalen Grid-Ansicht dieses Elements zu wechseln, wählen Sie die Tabelle oder eine ihrer Zeilen oder Spalten aus und

klicken Sie in der Symbolleiste auf das Symbol . Daraufhin wird das Tabellenelement in der Grid-Ansicht angezeigt.

Zeile einfügen

Umschalt+F12

Der Befehl **Zeile einfügen** ist in der Datenbank/Tabellen-Ansicht aktiv, wenn eine Zeile oder Zelle ausgewählt ist. Der Befehl fügt eine neue Zeile vor der markierten Zeile ein. Die neue Zeile entspricht einer Instanz des Tabellenelements. Für das neue Element werden obligatorische Child-Elemente erstellt.

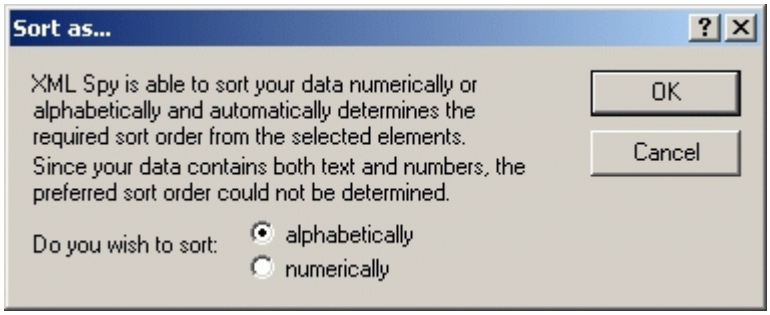
Zeile anhängen

Strg+F12

Der Befehl **Zeile anhängen** ist in der Datenbank/Tabellen-Ansicht aktiv, wenn eine Zeile oder Zelle ausgewählt ist. Der Befehl hängt eine neue Zeile nach der letzten Zeile der Tabelle an. Die neue Zeile entspricht einer Instanz des Tabellenelements. Für das neue Element werden obligatorische Child-Elemente erstellt.

Aufsteigend sortieren

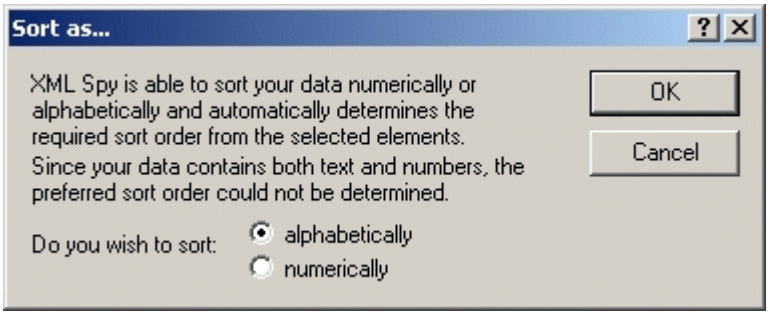
Der Befehl **Aufsteigend sortieren** ist in der Datenbank/Tabellen-Ansicht aktiv, wenn eine Spalte oder Zelle ausgewählt ist. Der Befehl sortiert die Spalte entweder alphabetisch oder numerisch in aufsteigender Reihenfolge. XMLSpy versucht automatisch zu ermitteln, um welche Art von Daten es sich handelt und sortiert diese alphabetisch oder numerisch. Wenn vom System nicht erkannt wird, um welche Art es sich handelt, werden Sie aufgefordert die Sortiermethode anzugeben (*siehe Abbildung unten*).



Absteigend sortieren

Der Befehl **Absteigend sortieren** ist in der Datenbank/Tabellen-Ansicht aktiv, wenn eine Spalte oder Zelle ausgewählt ist. Der Befehl sortiert die Spalte entweder alphabetisch oder numerisch in absteigender Reihenfolge. XMLSpy versucht automatisch zu ermitteln, um welche Art von Daten es sich handelt und sortiert diese alphabetisch oder numerisch. Wenn vom System nicht erkannt wird, um welche Art es sich handelt, werden Sie aufgefordert die Sortiermethode

anzugeben (*siehe Abbildung unten*).



15.4.6 Höher stufen

Strg+L

Der Befehl **Höher stufen** steht nur in der Grid-Ansicht zur Verfügung. Sie verschieben damit den markierten Node um eine Ebene nach links, d.h. ein Child-Element wird hierdurch zu einem Geschwister-Element seines Parent. Dieser Befehl wird auch "**Promote"** genannt.

15.4.7 Tiefer stufen

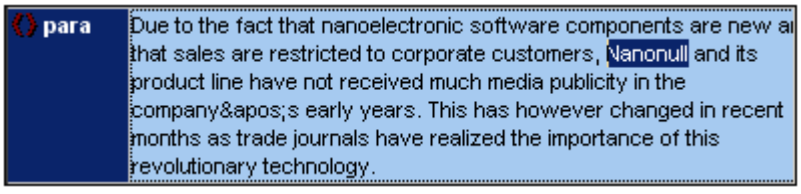
Strg+R

Der Befehl **Tiefer stufen** steht nur in der Grid-Ansicht zur Verfügung. Sie verschieben damit den markierten Node um eine Ebene nach rechts, sodass er hierdurch zu einem Child-Element seines ehemaligen Geschwister-Element wird. Dieser Befehl wird auch "**Demote"** genannt.

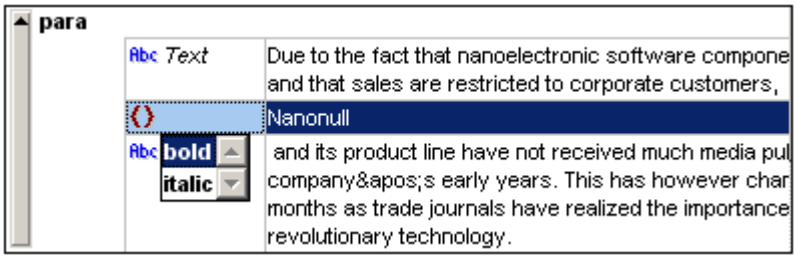
15.4.8 In Element einbinden

Der Befehl **In Element einbinden** steht nur in der Grid-Ansicht zur Verfügung. Er bindet einen markierten Textbereich in ein neues Element ein. Das neue Element wird innerhalb des Textbereichs rund um den markierten Text angelegt. Wenn Sie ein Dokument auf Basis eines Schemas oder einer DTD bearbeiten, steht Ihnen automatisch eine Liste mit gültigen Auswahlmöglichkeiten für den Namen des Elements, in das der Text eingeschlossen werden soll, zur Verfügung.

In der unten gezeigten Abbildung ist z.B. der Text `Nanonull` im Element `para` markiert.



Wenn Sie den Befehl **XML | In Element einbinden** auswählen, wird der Text "Nanonull" in ein neues Element eingeschlossen, das innerhalb des Texts erstellt wird. Es wird eine Liste angezeigt, aus der Sie für den Elementnamen `bold` oder `italic` auswählen können. Diese Elemente werden im Schema als Child-Elemente von `para` definiert.



Der gewählte Eintrag wird als Name des neuen Elements verwendet. Alternativ dazu können Sie auch einen anderen Namen für das Element eingeben.

15.4.9 XPath auswerten

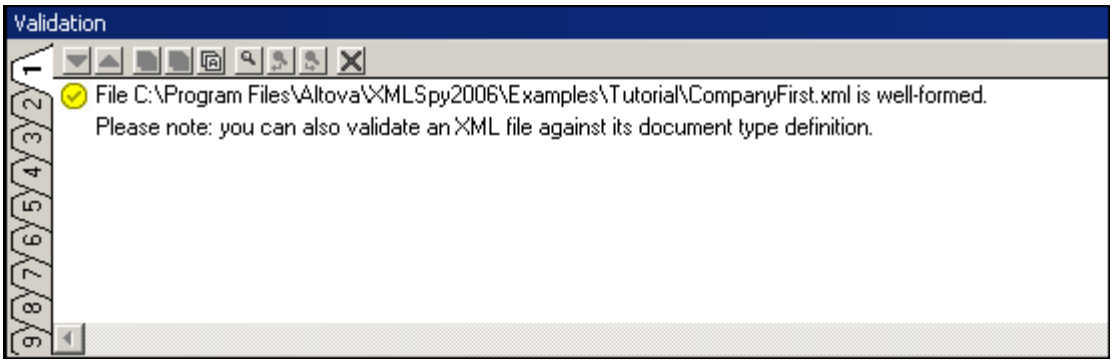
Bei Auswahl des Befehls **XML** | **XPath auswerten...** werden die Ausgabefenster geöffnet, wenn Sie noch nicht offen sind und das Register "XPath" wird in den Ausgabefenster aktiviert. Auf dem Register "XPath" können Sie einen XPath-Ausdruck im aktiven Dokument auswerten und das Ergebnis im Fenster "Ausgabe" anzeigen.

15.4.10 Wohlgeformtheit prüfen

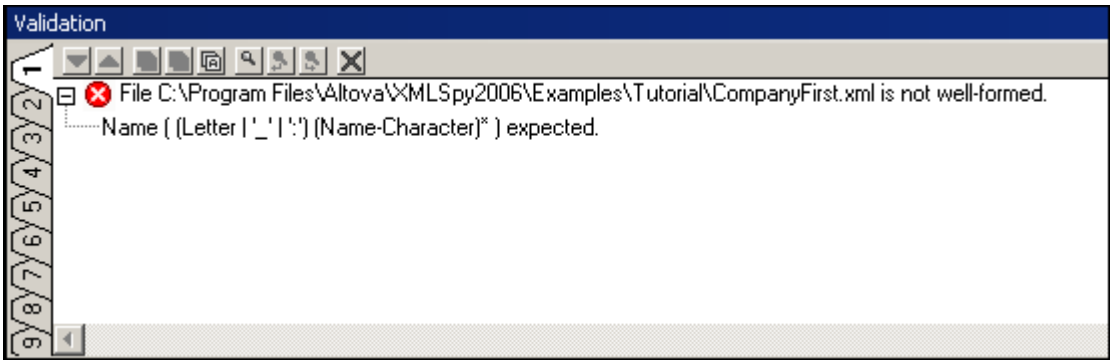
F7

Mit diesem Befehl überprüfen Sie das Dokument auf Wohlgeformtheit nach der Definition der XML 1.0 Spezifikation. Jedes XML-Dokument muss wohlgeformt sein; deswegen überprüft XMLSpy ein Dokument automatisch auf Wohlgeformtheit, sobald es geöffnet oder gespeichert wird, oder wenn eine andere als die Text-Ansicht ausgewählt wird. Sie können das Dokument auch während der Bearbeitung jederzeit auf Wohlgeformtheit prüfen.

Bei erfolgreicher Wohlgeformtheitsprüfung wird am unteren Rand des Hauptfensters eine Meldung angezeigt:



Wenn bei der Wohlgeformtheitsprüfung ein Fehler entdeckt wurde, wird das Fehler verursachende Objekt markiert und eine entsprechende Fehlermeldung am unteren Rand des Hauptfensters angezeigt.



Beachten Sie, dass das Validierungsfenster neun Ausgaberegister enthält. Das Ergebnis der Validierung wird immer auf dem aktuellen Register angezeigt. Sie können daher die Wohlgeformtheit für eine Schemadatei auf Register 1 überprüfen und das Ergebnis aufbewahren, indem Sie zu Register 2 wechseln, bevor Sie das nächste Schema-Dokument validieren (andernfalls wird Register 1 durch das Validierungsergebnis überschrieben.)

Eigentlich ist es nicht erlaubt, ein schlecht geformtes XML-Dokument zu speichern; XMLSpy enthält die Funktion "Save anyway" (trotzdem speichern), die angezeigt wird, wenn die automatische Wohlgeformtheitsprüfung beim Speichern des Dokument einen Fehler entdeckt. Dies ist nützlich, wenn sie Ihre Arbeit kurzzeitig unterbrechen müssen, und den Fehler nicht sofort beheben können.

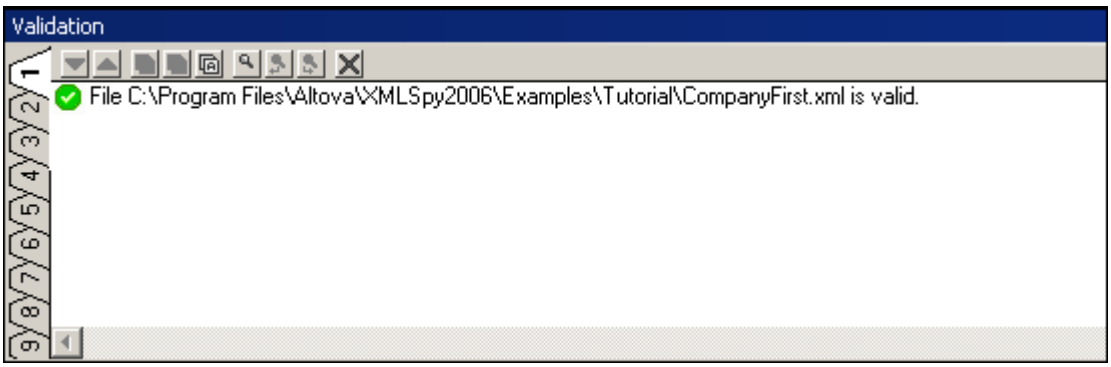
Bitte beachten Sie: Sie können den Befehl **Wohlgeformtheit prüfen** auch an jeder Datei, jedem Ordner oder jeder Gruppe von Dateien im aktiven Projektfenster anwenden. Klicken Sie auf den entsprechenden Ordner, und anschließend auf das Symbol "Wohlgeformtheit prüfen".

15.4.11 Validieren

F8

Mit diesem Befehl können Sie ein XML-Dokument gegen DTDs, XML-Schemas und andere Schemas validieren. Die Validierung wird automatisch durchgeführt, wenn Sie von der Textansicht in eine andere Ansicht wechseln. Sie können einstellen, dass ein Dokument beim Öffnen oder Speichern automatisch validiert wird (**Extras | Optionen | Datei**). Der Befehl **Validieren** führt vor Überprüfung der Gültigkeit gleichzeitig eine Wohlgeformtheitsprüfung durch, sodass diese vorher nicht separat überprüft werden muss.

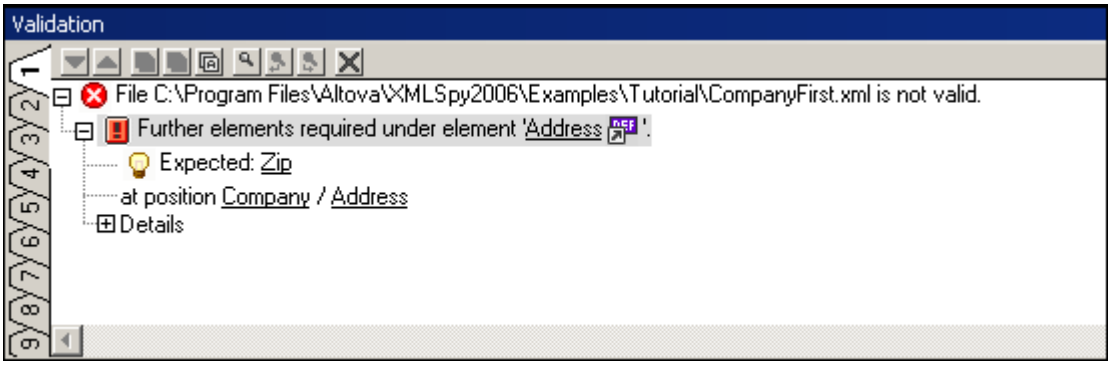
Bei erfolgreicher Validierung wird am unteren Rand des Hauptfensters folgende Meldung angezeigt:



Wenn bei der Validierung ein Fehler entdeckt wurde, wird das Fehler verursachende Objekt

markiert, und eine entsprechende Fehlermeldung wird angezeigt:

Bitte beachten Sie: Wenn Sie auf die Links in der Fehlermeldung klicken, können Sie zu der Stelle in der XML-Datei springen, an der der Fehler gefunden wurde.



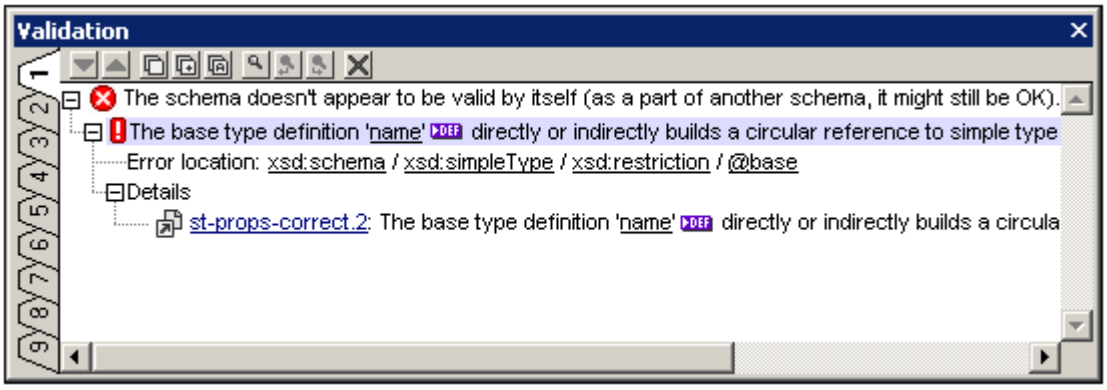
Bitte beachten Sie: Das Validierungsfenster enthält neun Ausgaberegister. Das Ergebnis der Validierung wird immer auf dem aktuellen Register angezeigt. Sie können daher die Wohlgeformtheit für eine Schemadatei auf Register 1 überprüfen und das Ergebnis aufbewahren, indem Sie zu Register 2 wechseln, bevor Sie das nächste Schema-Dokument validieren (andernfalls wird Register 1 durch das Validierungsergebnis überschrieben.)

Der Befehl wird normalerweise am aktiven Dokument durchgeführt. Sie können die Validierung über Dateien, Ordner oder Dateigruppen laufen lassen, die im aktiven Projektfenster angezeigt werden. Klicken Sie im Projektfenster auf die gewünschte Datei/den gewünschten Ordner und wählen Sie die Option **XML | Validieren** oder **F8.** Ungültige Dateien werden als aktive Datei im Hauptfenster geöffnet und die entsprechende Fehlermeldung wird angezeigt.

Validieren von XML-Dokumenten
Um eine XML-Datei zu validieren, machen Sie das XML-Dokument im Haupfenster zum aktiven und klicken Sie auf **XML | Validieren** oder drücken Sie **F8**. Das XML-Dokument wird daraufhin gegen das in der XML-Datei referenzierte Schema validiert. Existiert keine Referenz, wird am unteren Rand des Hauptfensters eine Fehlermeldung angzeigt. Solange das XML-Dokument geöffnet ist, bleibt das Schema im Speicher (siehe Cache-Speicher leeren im DTD/Schema-Menü).

Validieren von Schema-Dokumenten (DTDs und XML-Schema)
XMLSpy unterstützt die gebräuchlichen Schemadialekte einschließlich DTD und XML Schema. Um ein Schema-Dokument zu validieren, machen Sie es im Hauptfenster aktiv und klicken Sie auf **XML | Validieren** oder drücken Sie **F8**.

Schema-Validierung im Fenster "Validierung"
Wenn das Schema (DTD oder XML-Schema) gültig ist, wird im Validierungsfenster eine entprechende Meldung angezeigt; das Validierungsfenster wird unterhalb vom Applikationsfenster eingeblendet. Wenn das Schema nicht gültig ist, werden eine oder mehrere Fehlermeldungen im Validierungsfenster angezeigt (*Abbildung unten*).



Die Fehlermeldung ist in drei Teile gegliedert:

1. Eine Beschreibung des Fehlers, die Links zu den entsprechenden Deklarationen oder Definitionen enthält.
2. Wo in der Schemastruktur sich der Fehler befindet. Wenn Sie in der Pfadangabe auf einen Node klicken, wird dieser Node im Dokument markiert angezeigt.
3. Unter "Details" werden nähere Informationen zum Fehler angezeigt. Dieser Teil enthält einen Link zum entsprechenden Absatz in der jeweiligen Spezifikation (der XML-Spezifikation für DTD-Validierung; und der XML-Schema-Spezifikation für XML-Schema-Validierung).

Bitte beachten Sie: Wenn Sie bei Ausführung der Validierung in der Text-Ansicht im Validierungsfenster auf einen Link klicken, wird die entsprechende Deklaration oder Definition in der Text-Ansicht markiert. Wenn die Validierung in der Schema/WSDL-Ansicht durchgeführt wird, wird die entsprechende Deklaration oder Definition in der grafischen Schema/WSDL-Ansicht oder in der jeweiligen Eingabehilfe markiert.

Kataloge

XMLSpy unterstützt einen Teil der OASIS XML-Kataloge. Mit Hilfe von Katalogen kann XMLSpy gebräuchliche Schemas (sowie Stylesheets und andere Dateien) aus lokalen Benutzerordnern abrufen, wodurch die Verarbeitungsgeschwindigkeit erhöht wird, Benutzer auch offline arbeiten können (d.h. ohne mit einem Netzwerk verbunden zu sein) und Dokumente leichter übertragbar werden (da die URIs nur in den Katalogdateien geändert werden müssen). Der Mechanismus funktioniert folgendermaßen:

- XMLSpy lädt eine Datei namens `RootCatalog.xml`, die eine Liste von Katalogdateien enthält, die durchsucht werden. Sie können jeweils im Elemnt `nextCatalog` in `RootCatalog.xml` beliebig viele Katalogdateie definieren, die durchsucht werden sollen.
- Die in `RootCatalog.xml` inkudierten Katalogdateien werden durchsucht und die URIs werden entsprechend der Zuordnungen in den Katalogdateien aufgelöst. Achten Sie darauf, keine Zuordnungen doppelt zu vergeben, da dies Fehler verursachen könnte.
- Es werden mit XMLSpy (zusätzlich zu `RootCatalog.xml`) zwei Katalogdateien zur Verfügung gestellt: (i) `CoreCatalog.xml` (ordnet die vordefinierten `PUBLIC` System Identifier mehrerer gängiger Schemas (wie z.B. SVG und DocBook) URIs zu, die auf das/die lokal gespeicherte(n) Schemas verweisen und (ii) `CustomCatalog.xml`, eine Gerüstdatei, in der Sie Ihre eigenen Zuordnungen erstellen können). `RootCatalog.xml`, `CoreCatalog.xml` sowie `CustomCatalog.xml` sind in Ihrem XMLSpy Applikationsordner installiert. Die Schemas, Stylesheets und anderen Dateien, die von `CoreCatalog.xml` referenziert werden, befinden sich in spezifischen Unterordnern Ihres XMLSpy Installationsordners.
- Der `PUBLIC` oder der `SYSTEM` Identifier in der `DOCTYPE`-Anweisung Ihrer XML-Datei

 © 2007 Altova GmbH

wird für das Durchsuchen des Katalogs verwendet. Bei häufig verwendeten Schemas ist der `PUBLIC` Identifier normalerweise bereits vordefiniert, sodass nur mehr die URI in den Katalog-Lookup-Dateien geändert werden muss, wenn XML-Dokumente auf mehreren Rechnern verwendet werden.

Wenn Sie Ihre eigene `CustomCatalog.xml` (oder eine andere benutzerdefinierte Katalodatei) Datei schreiben, verwenden Sie nur die folgende Teilmenge des OASIS-Katalogs, damit XMLSpy den Katalog korrekt verarbeiten kann. Jedes der Elemente in der unterstützten Teilmenge kann das Attribut `xml:base` erhalten, welches dazu dient, die Basis-URI dieses Elements zu definieren.

```
<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>
```

Bitte beachten Sie:

- Die DTDs für XML-Schemas werden zwar von `CoreCatalog.xml` aus referenziert, doch können Sie Ihre XML-Dateien gegen keines dieser Schemas validieren. Zweck dieser beiden DTDs ist es, zu Bearbeitungszwecken für die Eingabehilfen Informationen bereitzustellen, falls Sie Dateien gemäß diesen älteren Empfehlungen erstellen wollen. *siehe auch nächster Punkt*
- Wenn Sie eine benutzerdefinierte Dateierweiterung für ein bestimmtes Schema erstellen (z.B. die Erweiterung `.myhtml` für (HTML)-Dateien, die gemäß der HTML-DTD gültig sein sollen), können Sie die intelligente Editierung für Dateien mit dieser Erweiterung aktivieren, indem Sie eine Textzeile zu `CustomCatalog.xml` hinzufügen. Für das erwähnte Beispiel sollten Sie das Element `<spy:fileExtHelper ext="myhtml" uri="schemas/xhtml/xhtml1-transitional.dtd"/>` als Child des Elements `<catalog>` hinzufügen, damit die intelligente Editierung (Autokomplettierung, Eingabehilfen usw.) von `.myhtml`-Dateien in XMLSpy gemäß der XHTML 1.0 Transitionsl DTD aktiviert werden kann.
- Nähere Informationen zu Katalogen finden Sie in der XML Catalogs specification.

Automatisches Validieren mit Altova XML 2007

Altova XML ist eine kostenlose Applikation, die den Altova XML-Validator sowie einen XLST 1.0-, XSLT 2.0- und XQuery 1.0-Prozessor enthält. Das Tool kann über die Befehlszeile, über eine COM-Schnittstelle, in Java-Programmen und in .NET-Applikationen zur Validierung von XML-Dokumenten, zum Transformieren von XML-Dokumenten unter Verwendung von XSLT 1.0 und 2.0 Stylesheets sowie zum Ausführen von XQuery-Dokumenten verwendet werden.

Validierungen können daher mittels Altova XML automatisiert werden. So können Sie z.B. eine Batch-Datei erstellen, die Altova XML aufruft, um damit eine Reihe von XML-Dokumenten zu validieren und die Ausgabe an eine Textdatei zu senden. Nähere Informationen dazu finden Sie in der Altova XML Dokumentation.

15.4.12 Validieren von WSDL-Dateien

Die folgende Liste enthält wichtige Informationen über die Validierung von WSDL-Dateien in der Enterprise und der Professional Edition von XMLSpy:

- Die Professional Edition führt nur eine einfache Schemavalidierung durch, d.h. die WSDL-Datei wird als XML-Datei behandelt und dem unter http://schemas.xmlsoap.org/wsdl/ definierten Schema entsprechend validiert.
- Die Enterprise Edition bietet Funktionen zur Validierung von WSDL-Dateien, die über den Umfang der Valdierungsfunktion in der Professional Edition hinausgehen. Die Datei wird nicht gegen http://schemas.xmlsoap.org/wsdl/ validiert, sondern gegen http://www.altova.com/specs_wsdl.html#_document-s, wobei zusätzlich mehr Validierungsinformationen im Zusammenhang mit WSDL berücksichtigt werden. Dadurch kann es vorkommen, dass eine WSDL-Datei zwar in der Professional Edition gültig ist, nicht aber in der Enterprise Edition (*siehe Beispiel*).
- Es besteht ein Unterschied zwischen http://schemas.xmlsoap.org/wsdl/ und http://www.altova.com/specs_wsdl.html#_document-s. In http://schemas.xmlsoap.org/wsdl/ fehlen die Extensibility-Elemente (die Spezifikation und das Schema sollten identisch sein, sind es aber nicht; dieser Unterschied scheint ein Fehler im Schema zu sein).
- Da in der Professional Edition zur Validierung http://schemas.xmlsoap.org/wsdl/ verwendet wird, werden alle Extensibilty-Element als ungültig betrachtet. In der Enterprise Edition wurde http://www.altova.com/specs_wsdl.html#_document-s verwendet, in dem Extensibility-Elemente zulässig sind. Daher ist eine WSDL-Datei, die Extensibility-Elemente enthält, gültig, wenn sie in der Enterprise Edition validiert wird, und ungültig, wenn sie in der Professional Edition validiert wird.
- Dass die Datei in der Professional Edition ungültig ist, liegt daran, dass wir das offizielle Schema der W3C Working Group verwenden. Fehler in diesem Schema liegen außerhalb unseres Einflusses.

Beispiel

Das folgende Beispiel stammt aus einer WSDL-Datei. Beachten Sie das in der Datei deklarierte Element "getCityTime". Dieses Element wird irrtümlicherweise als "getCityTimes" referenziert. Die Enterprise Edition überprüft, ob referenzierte Elemente zuvor in der Datei deklariert wurden; die Professional Edition tut dies nicht. Diese Datei wäre in der Professional Edition gültig (vorausgesetzt der Rest der Datei ist gültig), nicht aber in der Enterprise Edition (es sei denn, "getCityTimes" ist an einer anderen Stelle in der Datei definiert).

```
<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>
```

 © 2007 Altova GmbH

15.4.13 Eingabehilfen aktualisieren

Mit diesem Befehl aktualisieren Sie die Eingabehilfen-Fenster. Dies erfolgt durch erneutes Laden der zugrunde liegenden DTD / des zugrunde liegenden Schemas.
Sollten Sie das Schema oder die DTD, auf dem/auf der ein offenes XML-Dokument basiert, verändert haben, empfiehlt es sich, das Schema oder die DTD erneut zu laden, damit die intelligenten Editierinformationen im XML-Dokument aktualisiert werden.

15.4.14 Namespace Präfix...

Dieser Befehl steht in der Grid-Ansicht zur Verfügung und öffnet ein Dialogfeld, in dem Sie das Namespace Präfix des ausgewählten Elements oder Attributs - und im Fall von Elementen - seiner Nachfahren definieren können.

Sie können das Namespace Präfix für Elemente, Attribute oder für beide setzen. Das Namespace Präfix wird auf das ausgewählte Element oder Attribut angewendet und - wenn ein Element ausgewählt ist - auch auf die untergeordneten Nodes des ausgewählten Elements.

15.5 Menü "DTD/Schema"

Das **DTD/Schema**-Menü enthält Befehle für das effiziente Arbeiten mit DTDs und XML-Schemas.

Dieser Abschnitt enthält eine umfassende Beschreibung aller Befehle in diesem Menü.

15.5.1 DTD zuweisen

Dieser Befehl ist aktiv, wenn eine XML-Datei aktiv ist. Sie weisen damit einem XML-Dokument eine DTD zu, wodurch Validierung und intelligentes Editieren ermöglicht wird. Wenn Sie diesen Befehl wählen, wird das Dialogfeld zum Zuweisen einer Datei geöffnet, in dem Sie die gewünschte DTD zuweisen können. Beachten Sie, dass Sie den Pfad der zugewiesenen Datei relativ machen können. Aktivieren Sie dazu das Kontrollkästchen "Pfad relativ zu.. machen". Nach erfolgter Zuweisung enthält Ihr Dokument eine DOCTYPE-Deklaration, die die zugewiesene DTD referenziert. Die DOCTYPE-Deklaration sieht ungefähr folgendermaßen aus:

```
<!DOCTYPE main SYSTEM"http://link.xmlspy.com/spyweb.dtd">
```

Beachten Sie, dass eine DTD einer neuen XML-Datei zum Zeitpunkt der Erstellung zugewiesen werden kann.

15.5.2 Schema zuweisen

Dieser Befehl ist aktiv, wenn eine XML-Datei aktiv ist. Sie weisen damit einem XML-Dokument ein XML-Schema zu, wodurch Validierung und intelligentes Editieren ermöglicht wird. Wenn Sie diesen Befehl wählen, wird das Dialogfeld zum Zuweisen einer Datei geöffnet, in dem Sie das gewünschte XML-Schema zuweisen können. Beachten Sie, dass Sie den Pfad der zugewiesenen Datei relativ machen können. Aktivieren Sie dazu das Kontrollkästchen "Pfad relativ zu... machen". Nach erfolgter Zuweisung enthält Ihr Dokument eine XML-Schema-Zuweisung mit dem benötigten Namespace. Die Schema-Zuweisung sieht ungefähr folgendermaßen aus:

```
      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"
```

Die mit XMLSpy erzeugten Deklarationen hängen vom Schematyp und von der potentiellen Verwendung eines targetNamespace im Schemadokument ab.

15.5.3 Weitere DTD einfügen

Mit diesem Befehl können Sie eine weitere Dokument Type Definition (DTD) oder extern geparste Entity in das interne Subset einer DTD, oder in jede andere DTD inkludieren. Dies geschieht durch Definition einer entsprechenden extern geparsten Entity-Deklaration und Verwendung dieser Entity in folgender Zeile:

```
<!ENTITY % navigation.dtd SYSTEM"S:\xml\navigation.dtd">
%navigation.dtd;
```

 © 2007 Altova GmbH

Mit diesem Befehl öffnen Sie das Dialogfeld zum Zuweisen einer Datei, in dem Sie angeben können, welche DTD in Ihre DTD inkludiert werden soll.

Bitte beachten Sie: Dieser Befehl ist nur in der Grid-Ansicht aktiv.

15.5.4 Gehe zu DTD

Mit diesem Befehl öffnen Sie die DTD, auf der das aktive XML-Dokument basiert. Wurde keine DTD zugewiesen, wird eine Fehlermeldung angezeigt.

15.5.5 Gehe zu Schema

Mit diesem Befehl öffnen Sie das XML-Schema, auf dem das aktive XML-Dokument basiert. Wurde kein XML-Schema zugewiesen, wird eine Fehlermeldung angezeigt.

15.5.6 Gehe zu Definition

Mit diesem Befehl wird die exakte Definition eines Elements oder Attributs in der zugehörigen DTD bzw. dem zugehörigen Schema-Dokument angezeigt.

So zeigen Sie die Definition des Datenelements in der Enhanced Grid Ansicht an:

1. Klicken Sie auf das Datenelement.
2. Wählen Sie im Menü **DTD/Schema | Gehe zu Definition**, oder klicken Sie auf das oben abgebildete Symbol.

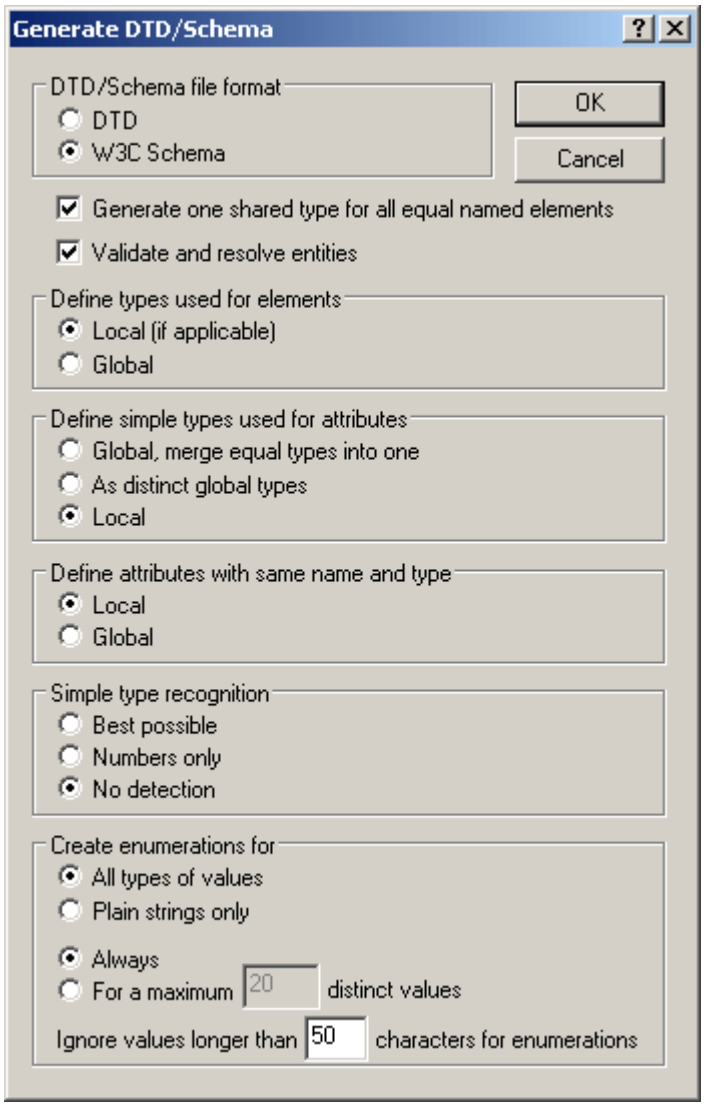
So zeigen Sie die Definition des Datenelements in der Schema/WSDL Design-Ansicht an:

1. Mit Strg + Doppelklick auf das Datenelement, dessen Definition Sie ansehen möchten, oder
2. Klicken Sie auf das Objekt und wählen Sie die Menüoption **DTD/Schema | Gehe zu Definition,** oder klicken Sie auf das Symbol.

   In beiden Fällen wird die zugehörige DTD- oder Schemadatei geöffnet und die Definition des Datenelements wird markiert.

15.5.7 DTD/Schema generieren

Mit diesem Befehl wird eine neue DTD oder ein W3C XML-Schema von einem XML-Dokument (oder von einer Gruppe von Dokumenten im Projekt Fenster) erzeugt.
Verwenden Sie diese Funktion, wenn Sie ein neues Schema oder eine neue DTD anhand eines XML-Dokuments erzeugen möchten.



Bei der Generierung eines XML-Schemas stehen die folgenden Optionen zur Verfügung:

- ***Elemente***: Die Art der Elemente kann lokal oder global definiert werden (*für Elemente verwendeten Typ definieren*). Wenn Elemente denselben Namen haben, kann ein allgemeiner Typ definiert werden, der in der Definition dieser Elemente verwendet wird (*für alle gleichnamigen Elemente einen gemeinsamen Typ erstellen)*.
- ***Attribute***: Die simpleTypes von Attributen (*für Attribute verwendete simpleTypes definieren*) können definiert werden als: (i) allgemeine globale Typen; (ii) unterschiedliche globale Typen; (iii) lokale Typen. Attribute desselben Namens und Typs können entweder lokal oder global definiert werden.
- ***SimpleType-Erkennung***: Die Typerkennung (*SimpleType-Erkennung*) kann folgendermaßen festgelegt werden: (i) als bestmögliche; (ii) nur Zahlen-Datentypen werden erkannt; (iii) keine Datentyp-Erkennung. In diesem Fall werden alle Datentypen auf `xs:string` gesetzt.
- ***Entity-Auflösung***: Im XML-Dokument können Entities im Elementinhalt und in Attributwerten vorkommen. Es ist daher für Enumerationswerte von Bedeutung, ob diese Entities aufgelöst werden sollen oder nicht (*Entities validieren und auflösen*). Außerdem können sich einige Entities (v.a. geparste Entities, die Markup-Tags enthalten) unterschiedlich auf das Content Model auswirken, je nachdem, ob sie aufgelöst werden oder nicht. Beachten Sie, dass das XML-Dokument auf seine

 © 2007 Altova GmbH

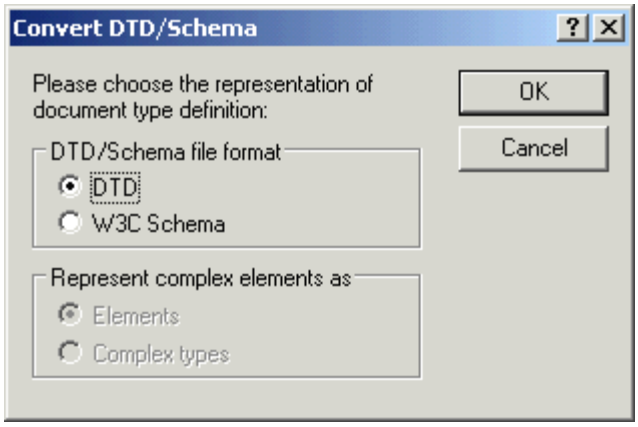
korrekte XML-Syntax valdiert wird, bevor das Schema generiert wird. Wenn das Dokument ungültig ist, wird die Schemagenerierung abgebrochen.
- ***Enumerationen***: Es können Enumerationen für alle Wertetypen oder nur für Stringwerte erstellt werden.

Bei Generierung einer DTD stehen die Optionen für die Entity-Auflösung und Enumerationen zur Verfügung.

Der Befehl "DTD/Schema generieren" wird normalerweise im aktiven Hauptfenster durchgeführt, allerdings können Sie ihn auch auf jede Datei, jeden Ordner oder jede Gruppe von Dateien im aktiven Projekt Fenster anwenden.

15.5.8 DTD/Schema konvertieren

Mit diesem Befehl konvertieren Sie eine existierende DTD in ein XML-Schema und umgekehrt. Bei Auswahl der Option "W3C Schema" müssen Sie außerdem angeben, wie Complex-Elemente dargestellt werden sollen - als Elemente oder complexTypes.



Kovertieren einer DTD in ein XML-Schema
Bei der Konvertierung einer DTD in ein XML-Schema macht XMLSpy aufgrund der beschränkten Informationen, die zur Verfügung stehen, einige Annahmen. Beachten Sie vor allem, dass die Werte bestimmter DTD-Komponenten wörtlich interpretiert werden, anstatt dass ihre Semantik geparst wird, da das Programm nicht weiß, welche der vielen Möglichkeiten die richtige ist. Bearbeiten Sie die generierte Konvertierung in diesem Fall.

Das Resultat sollte jedenfalls immer genau überprüft und gegebenenfalls verbessert werden. Im Folgenden werden einige Bereiche angegeben, in denen Sie eventuell Korrekturen vornehmen sollten.

Definition von Datentypen für Attribute
DTDs unterstützen nur 10 Attribut-Datentypen, während in XML-Schemas z.B. 44 Datentypen plus davon abgeleitete Datentypen zulässig sind. Durch Verwendung von weniger Datentypen lässt sich ein generiertes XML-Schema unter Umständen verbessern. Beachten Sie, dass Datentyp-Informationen bei der Konvertierung eines XML-Schemas in eine DTD verloren gehen.

Namespaces
DTD erkennen keine Namespaces. Wenn daher in einer DTD Namespaces definiert werden, müssen sie in Form von Element- und Attributnamen hartkodiert werden. Bei der Konvertierung

von einem Schema in ein anderes kann dies einige Probleme verursachen.

Entities
In XML-Schemas gibt es kein Äquivalent für die allgemeinen Entity-Deklarationen von DTDs. Wenn XMLSpy eine DTD in ein XML-Schema konvertiert, ignoriert es die Entity-Deklarationen.

Nicht geparste Datendeklarationen
DTDs und XML-Schemas verwenden bei der Behandlung nicht geparster Daten unterschiedliche Mechanismen. Dies wird im Folgenden näher erklärt.

DTDs verwenden den folgenden Mechanismus:

1. Eine Notation wird aus einem Namen und einem Identifier bestehend deklariert, z.B.
   ```
   <! NOTATION gif SYSTEM "image/gif">
   ```
2. Sie deklarieren die Entity, z.B.
   ```
   <! ENTITY cover_img SYSTEM "graphics/cover_img.gif" NDATA gif>
   ```
3. Normalerweise gibt man zum betreffenden Attribut einen Attributtyp von ENTITY an, z.B.
   ```
   <! ELEMENT img EMPTY>
   <! ATTLIST img format ENTITY #REQUIRED>
   ```

In einem XML-Schema wird hierfür der folgende Mechanismus verwendet:

1. Deklarieren Sie eine Notation. Dies funktioniert auf dieselbe Art wie bei der DTD.
   ```
   <xs:notation name="gif" public="image/gif"/>
   ```
   Beachten Sie, dass das Attribut `public` obligatorisch ist und den Identifier enthält. Ein optionales Attribut `system` enthält den System Identifier und ist normalerweise eine ausführbare Datei, die Ressourcen des Notationstyps verarbeiten kann.
2. Man verknüpft die Notationsdeklaration über den NOTATION-Datentyp mit einem bestimmten Attributwert.
   - Sie können den `NOTATION`-Datentyp jedoch nicht direkt verwenden, sondern müssen einen weiteren Datentyp aus dem NOTATION-Datentyp ableiten.
     ```
     <xs:simpleType name="formatType">
       <xs:restriction base="xs:NOTATION">
         <xs:enumeration value="gif"/>
         <xs:enumeration value="jpeg"/>
       </xs:restriction>
     </xs:simpleType>
     ```
   - Man verknüpft das Attribut mit dem vom `NOTATION`-Datentyp abgeleiteten Datentyp, z.B.
     ```
     <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"/>
     ```

Beim Konvertieren einer DTD in ein XML-Schema mit Hilfe des Befehls "DTD/Schema konvertieren..." geht XMLSpy vor wie folgt:

- Die folgende Deklaration:
  ```
  <! ATTLIST image format ENTITY #REQUIRED
  ...>
  ```
  wird in

 © 2007 Altova GmbH

`<xs:attribute name="format" type="xs:ENTITY" use="required"/>`
konvertiert.

- und
  `<!NOTATION gif SYSTEM "image/gif">`

  wird in
  `<xs:notation name="gif" system="image/gif"/>`
  konvertiert.

Nehmen Sie daher die folgenden Änderungen vor:

1. In Notationen wie `<xs:notation name="gif" system="image/gif"/>` ersetzen Sie `system` durch `public` und fügen einen optionalen System Identifier hinzu, falls erforderlich.
2. Leiten Sie aus dem `NOTATION` Datentyp einen Datentyp ab, wie oben für `formatType` beschrieben.
3. Verknüpfen Sie den abgeleiteten Datentyp mit dem entsprechenden Attribut.

Bitte beachten Sie: Gemäß der XML-Schemaspezifikation, ist es nicht notwendig - bzw. je nach Sichtweise, nicht möglich - eine externe Entity zu deklarieren.

Konvertieren eines XML-Schemas in eine DTD
Bei der Konvertierung eines XML-Schemas in eine DTD werden die im XML-Schema verwendeten **Namespace-Präfixe** - nicht die Namespace URIs oder die Namespace-Deklarationen - in die Namen der entsprechenden Elemente und Attribute in der DTD übernommen. Beachten Sie die folgenden Punkte:

1. Da XML-Parser Namespaces beim Validieren eines Dokuments gegen eine DTD ignorieren, werden die Namespace-Deklarationen selbst nicht konvertiert.
2. Die Attribute `elementFormDefault` und `attributeFormDefault` des Elements `xs:schema` bestimmen, von welchen Elementen und Attributen die Präfixe im Konvertierungsvorgang übernommen werden. Ist der Wert auf `unqualified` gesetzt, werden nur die Präfixe von global deklarierten Elementen bzw. Attributen im Konvertierungsprozess übernommen. Ist der Wert auf `qualified` gesetzt, werden die Präfixe aller Element- und Attributnamen in den Konvertierungsprozess eingeschlossen.
3. Präfixe werden in den entsprechenden Stringwert plus Doppelpunkt konvertiert. Elemente und Attribute in Default Namespaces werden in Elemente und Attribute konvertiert, deren Namen mit: `default_NS_X` beginnt, wobei `X` eine Ganzzahl ist (beginnend mit 1 bis zu einem Maximalwert, der der Anzahl der im XML-Schema verwendeten Default Namespaces entspricht).
4. In der DTD sind Elementnamen aus Parameter-Entities zusammengesetzt. Auf diese Art können Sie das Präfix in der DTD jederzeit ändern, falls das Präfix im XML-Dokument einmal geändert werden muss. Parameter-Entity-Definitionen können entweder in der DTD selbst oder durch Außerkraftsetzung der Parameter-Entity-Definitionen im internen DTD Subset des XML-Dokuments geändert werden.

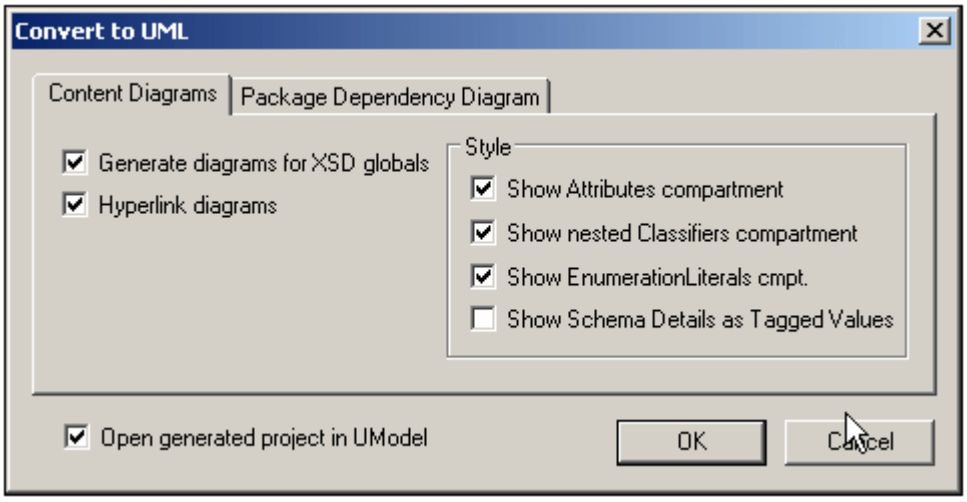
Bitte beachten Sie: Namespaces haben keinen semantischen Wert in einer DTD. Namespace-Präfixe, die aus dem XML-Schema übernommen werden, werden einfach nur Teil des Namens des in der DTD definierten Elements oder Attributs.

15.5.9 Im UML konvertieren

Mit dem Befehl **DTD/Schema | In UML konvertieren** wird ein W3C XML-Schema in ein Altova UModel-Projekt (. `ump`) Dokument (in Folgenden UModel-Projekt genannt) konvertiert. UMP ist das native Format von Altova UModel, der UML-Modellierungsapplikation von Altova. UMP-Dateien können in Altova UModel angezeigt und editiert werden.

Um ein Schema in UML zu konvertieren, gehen Sie folgendermaßen vor:

1. Klicken Sie, während das Schema geöffnet ist, auf den Befehl **In UML konvertieren**. Daraufhin erscheint das Dialogfeld "In UML konvertieren" (*Abbildung unten*).



2. Wählen Sie auf dem Register "Inhaltsdiagramme" die Option "Diagramm für globale XSD-Elemente generieren". Daraufhin wird im UModel-Projekt ein Content Model-Diagramm für jede globale Komponente erstellt.
3. Wählen Sie die erforderlichen Optionen aus den im Dialogfeld verfügbaren aus. Diese Optionen werden weiter unten beschrieben.
4. Wenn Sie das erstellte Projekt sofort in UModel ansehen möchten, wählen Sie die Option zum Öffnen des Projekts in UModel. Lassen Sie die Option andernfalls deaktiviert.
5. Klicken Sie auf **OK**.
6. Suchen Sie im Dialogfeld "Speichern unter", das nun angezeigt wird, den Zielordner, geben Sie anschließend den Namen der UMP-Datei an und klicken Sie auf **Speichern.**

Optionen für das Konvertieren in UML
Im Dialogfeld "In UML konvertieren" stehen die folgenden Optionen zur Verfügung.

Auf dem Register ***Inhaltsdiagramme***:

- *Diagramme verknüpfen* erstellt in jedem Diagramm einen Link zum Eintrag dieser globalen Komponente in der Modell-Strukturansicht, so dass Sie die Komponente in der Schemahierarchie schnell finden.
- Im Bereich *Stil* können Sie über die Optionen zur Anzeige der Bereiche verschiedene Bereiche ein oder ausblenden.

Auf dem Register ***Paketabhängigkeitsdiagramm***:

- Die Option *Diagramm generieren* legt fest, ob ein Paketabhängigkeitsdiagramm generiert werden soll. Ein Paketabhängigkeitsdiagramm bietet eine Übersicht über das gesamte Paket und zeigt die Beziehungen der Paketkomponenten zueinander.

© 2007 Altova GmbH

Beachten Sie, dass die anderen Optionen auf diesem Register nur aktiviert sind, wenn die Option "Diagramm generieren" aktiviert ist.

- Bei Auswahl der Option *Paket mit Diagramm verknüpfen* wird ein Link vom Paketdiagramm zur Modell-Strukturansicht erstellt.
- Für das Layout des Paketabhängigkeitsdiagramms stehen drei Optionen zur Verfügung: (i) nicht strukturiertes Layout (die Option "Automatisches Layout" ist deaktiviert); (ii) hierarchisches Layout (die Optionen "Automatisches Layout" und "hierarchisch" sind aktiviert); und (iii) zentriert (die Optionen "Automatisches Layout" und "Zentriert" sind aktiviert). Das Layout kann durch Editieren des Diagramms in UModel geändert werden..

Anmerkung: Die Funktion "In UML konvertieren" unterstützt nur W3C XML-Schemas.

15.5.10 In MapForce auf andere(s) DTD/Schema oder DB mappen...

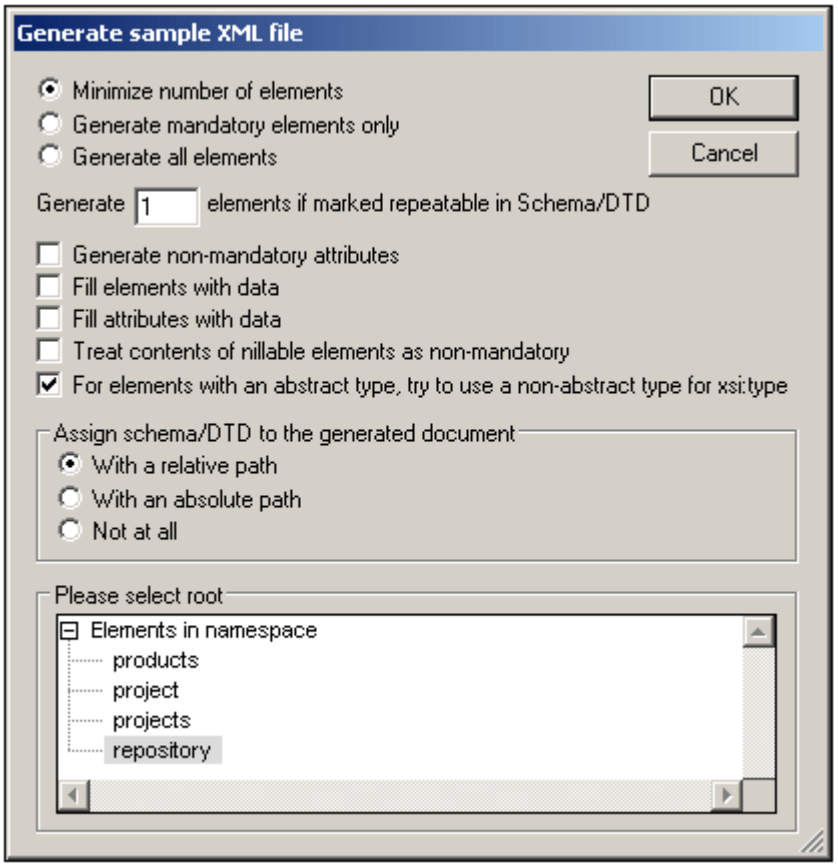
Startet Altova MapForce, wenn die Applikation installiert ist. In MapForce können Sie ein Schema auf eine andere DTD, ein anderes XML-Schema oder eine Datenbank mappen.

15.5.11 HTML/PDF-Ausgabe in StyleVision erstellen

Startet Altova StyleVision, wenn die Applikation installiert ist. In StyleVision können Sie Stylesheets für die Ausgabe von Dokumenten im HTML-, PDF- und RTF-Format erstellen.

15.5.12 XML Beispieldatei erzeugen

Dieser Befehl erstellt eine XML-Datei, die auf dem aktiven Schema (DTD oder XML-Schema) im Hauptfenster basiert.



Zu generierende Elemente
Sie haben eine der folgenden Auswahlmöglichkeiten: (i) nur obligatorische Elemente; (ii) alle Elemente (obligatorische und nicht obligatorische); (iii) obligatorische und Mindestanzahl nicht obligatorischer Elemente (z.B. die erste Definition in einer Auswahl). Wenn Elemente als wiederholend definiert sind, kann die Anzahl der zu generierenden Elemente (für jedes dieser wiederholenden Elemente) festgelegt werden.

Nicht obligatorische Attribute erzeugen
Beim Aktivieren dieser Option werden alle obligatorischen und optionalen Attribute, die im Schema definiert sind, generiert.

Erzeuge "X" Element(e), wenn als wiederholend definiert (Schema/DTD)
Beim Aktivieren dieser Option wird die im Textfeld eingegebene Anzahl von sich wiederholenden Elementen generiert.

Elemente und Attribute mit Daten befüllen
Beim Aktivieren dieser Option werden die Datentype-Deskriptoren/Werte der jeweiligen Elemente/Attribute eingefügt. Z.B.: `Boolean = 1`, `xsd: string = string`, Max/Min inklusive = der im Schema definierte Wert.

Nillable Elemente und abstrakte Typen
Der Inhalt von nillable Elemente kann als nicht obligatorisch behandelt werden und Elemente mit einem abstrakten Typ können einen nicht abstrakten Typ für ihr `xsi: type` Attribut verwenden.

Schemazuweisung für die generierte XML-Datei
Das für die Generierung der XML-Datei verwendete Schema kann der generierten XML-Datei

 © 2007 Altova GmbH

mit einem relativen oder einem absoluten Pfad zugewiesen werden.

Root-Element
Wenn das Schema mehr als ein globales Element enthält, werden diese aufgelistet und Sie können das für die XML-Beispieldatei benötigte Root-Element aus der Liste auswählen.

15.5.13 Programmcode generieren

Mit diesem Menüpunkt werden automatisch Java-, C++- oder C#-Klassendateien aus Definitionen im aktiven Schema (DTD oder XML-Schema) generiert.
Weitere Informationen dazu erhalten Sie in der Code Generator Dokumentation.

15.5.14 Cache-Speicher leeren

Mit diesem Befehl werden alle zwischengespeicherten Schema- und DTD-Dokumente aus dem Speicher gelöscht. Um Validierung und intelligentes Editieren zu beschleunigen, werden in XMLSpy alle zuletzt verwendeten Schemadokumente und externen geparsten Entities im Zwischenspeicher gespeichert. Informationen aus diesen Dokumenten werden auch mittels des Befehls Gehe zu Definition angezeigt.

Verwenden Sie diesen Befehl, wenn Sie nicht genügend Speicher in Ihrem System haben, oder wenn Sie zuletzt mit sehr vielen Dokumenten gearbeitet haben, die auf verschiedenen Schemas basieren.

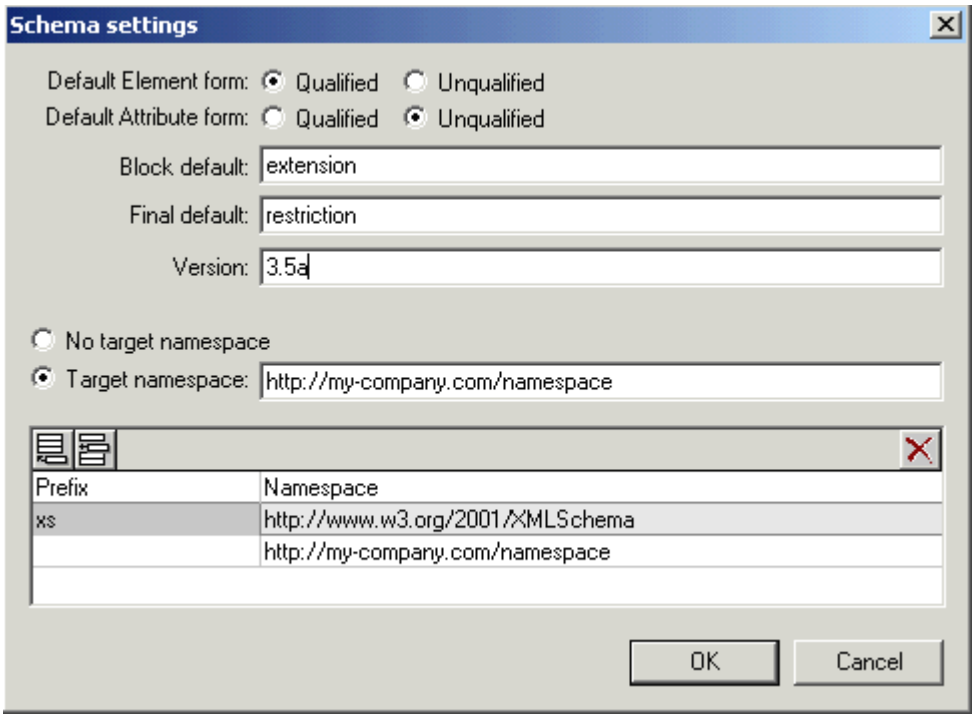
15.6 Menü "Schema-Design"

Im Schema-Design-Menü können Sie die Schema-Design-Ansicht von XMLSpy konfigurieren. In dieser Ansicht können Sie XML-Schemas über eine grafische Benutzeroberfläche erstellen. Das Menü steht dann zur Verfügung, wenn ein XML-Schemadokument in der Schema/WSDL-Ansicht aktiv ist.

Die Befehle in diesem Menü werden in diesem Abschnitt beschrieben.

15.6.1 Schema-Einstellungen

Mit diesem Befehl definieren Sie globale Einstellungen für das aktive Schema.
Dabei handelt es sich um Attribute und deren Werte für das XML-Schema-Dokument-Element `xs:schema`.



In diesem Dialogfeld können Sie die folgenden Einstellungen vornehmen:

- Sie können die Attribute `elementFormDefault` und `attributeFormDefault` des xs:schema-Elements jeweils auf `qualified` oder `unqualified` setzen.
- Weiters können Sie die Attribute `blockDefault`, `finalDefault` and `version` durch Eingabe der benötigten Werte in das jeweilige Textfeld definieren.
- Um einen Target Namespace für das XML-Instanzdokument zu definieren, aktivieren Sie das Target Namespace-Optionsfeld und geben Sie den Target Namespace in das Textfeld ein. Bei Deklaration eines Target Namespace müssen Sie diesen Namespace im Schema-Dokument durch Eingabe einer Namespace-Zeile im unteren Bereich definieren. Sie können für diesen Namespace ein Präfix definieren oder ihn als Default Namespace verwenden (indem Sie das Präfix-Feld wie in der obigen Abbildung gezeigt leer lassen).

© 2007 Altova GmbH

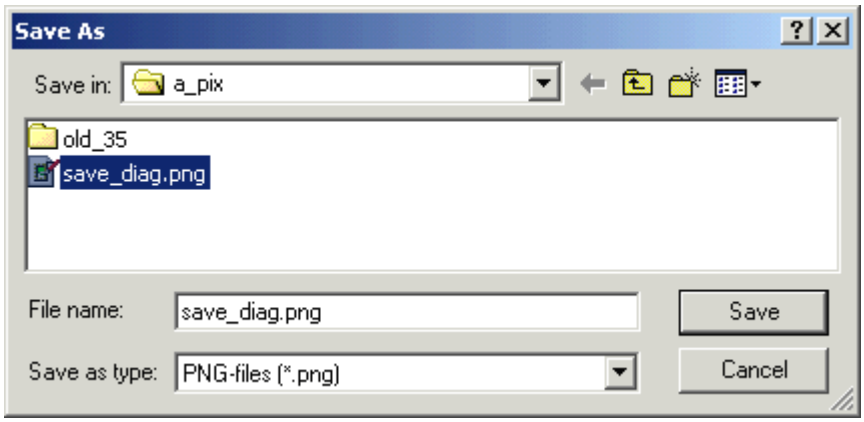
In der Textansicht sehen die obigen Schema-Einstellungen etwa folgendermaßen aus:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema
targetNamespace="http://www.xmlspy.com/schemas/Altova/orgc
hart" xmlns="http://www.xmlspy.com/schemas/Altova/orgchart"
xmlns:xsd="http://www.w3.org/2000/10/XMLSchema"
elementFormDefault="unqualified"
attributeFormDefault="unqualified" blockDefault="extension"
finalDefault="restriction" version="3.5a">
    <xsd:notation name="Altova-Orgchart"
public="http://www.xmlspy.com/schemas/Altova/orgchart"/>
    <xsd:complexType name="DivisionType">
        <xsd:sequence>
            <xsd:element name="Name" type="xsd:string">
                <xsd:annotation>
                    <xsd:documentation>Division Name
```

Bitte beachten Sie: Diese Einstellungen gelten nur für das aktive Dokument.

15.6.2 Diagramm speichern...

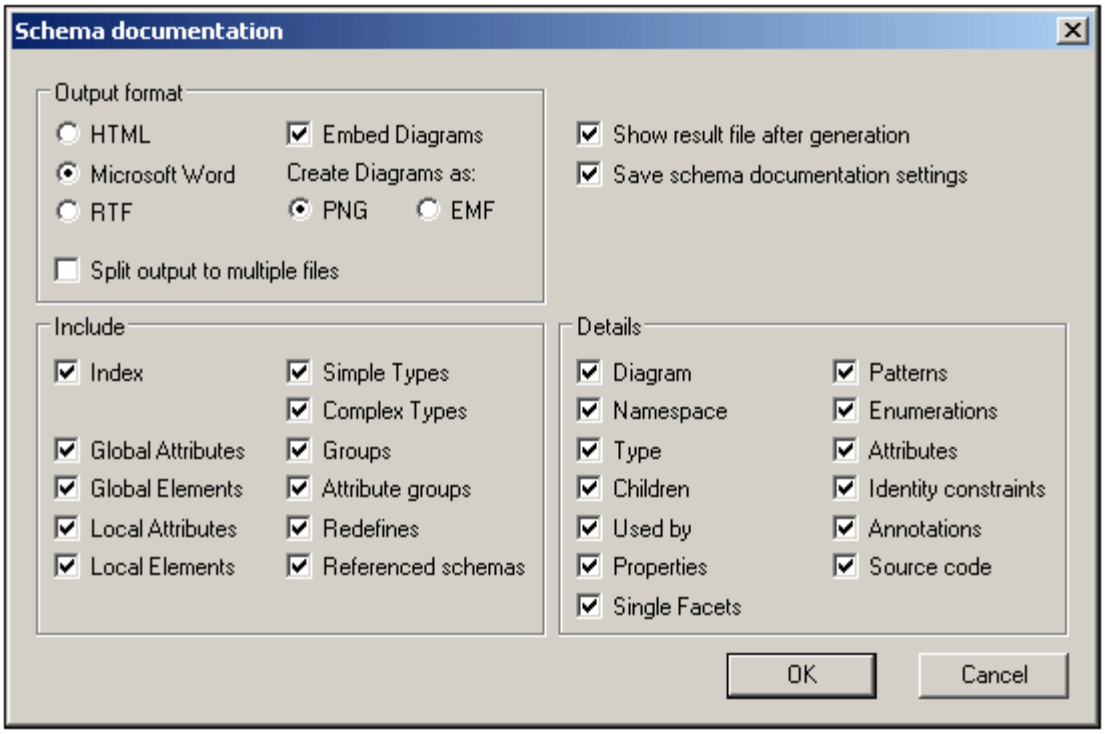
Mit diesem Befehl wird das Diagramm des im Hauptfenster angezeigten Content Model im PNG-Format unter dem gewünschten Pfad gespeichert.



15.6.3 Schema-Dokumentation generieren

Mit diesem Befehl wird eine detaillierte Dokumentation zum Schema im HTML-, MS Word oder RTF-Format erzeugt. Die Dokumentation wird für diejenigen Komponenten generiert, die Sie im Dialogfeld "Schema-Dokumentation" auswählen (Aufruf des Dialogfelds mit dem Befehl "Schema-Dokumentation generieren", *siehe Abbildung unten*). Verwandte Elemente (Child-Elemente, ComplexTypes, usw.) werden in der Bildschirmausgabe mit Hyperlinks versehen, sodass Sie zwischen den einzelnen Komponenten navigieren können. Komponenten mit einem Content Model weisen auch Links zur jeweiligen Content Model-Definition auf. Beachten Sie, dass Schema-Dokumentation auch für **inkludierte und importierte Schemakomponenten** generiert wird.

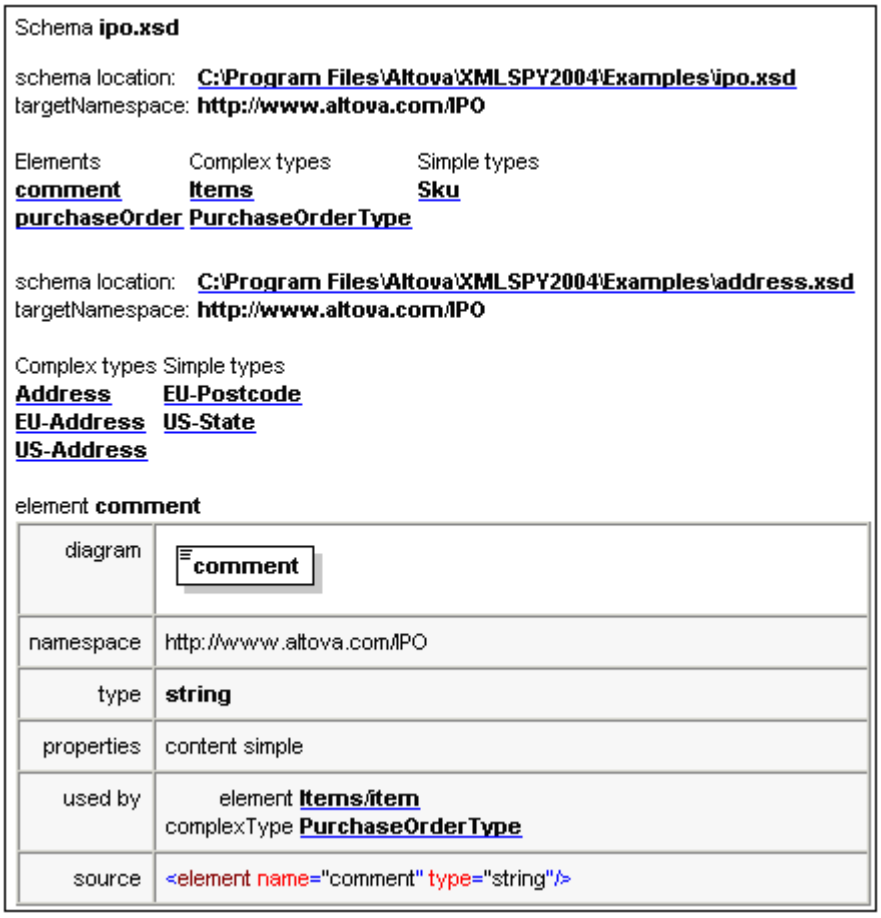
Anmerkung: Um Dokumentation im Format MS Word zu generieren, müssen Sie MS Word (Version 2000 oder höher) installiert haben.



Wenn Sie auf den Befehl "Schema-Dokumentation generieren" klicken, wird das Dialogfeld "Schema-Dokumentation" aufgerufen:

- Das gewünschte Format wird im Bereich "Output-Format" definiert: entweder HTML, MS Word oder RTF. Die Dokumentation kann entweder als eine einzige Datei generiert oder in mehrere Dateien aufgeteilt werden. Wenn mehrere Dateien generiert werden, entspricht jede Datei einer Komponente. Welche Komponenten in der Ausgabe enthalten sein sollen, wird mittels der Kotrollkästchen im Bereich "Inkludieren" festgelegt.
- Die Option "Diagramme einbetten" ist für die Ausgabeoptionen MS Word und RTF aktiv. Wenn diese Option aktiviert ist, werden Diagramme entweder im PNG- oder im EMF-Format in die Ergebnisdatei eingebettet. Andernfalls werden die Diagramme als PNG- oder EMF-Dateien generiert und in der Ergebnisdatei über Objektlinks angezeigt. Wenn es sich bei der Ausgabedatei um eine HTML-Datei handelt, werden alle Diagramme als dokumentexterne PNG-Dateien generiert.
- Im Bereich **Inkludieren** können Sie auswählen, welche Datenelemente in der Dokumentation inkludiert werden sollen. Bei Auswahl der Option "Index" werden alle verwandten Schemas mit ihren globalen Komponenten nach Komponententyp geordnet am Anfang der Datei aufgelistet.
- Im Bereich **Details** werden die Einzelheiten aufgelistet, die für die einzelnen Komponenten inkludiert werden können. Wählen Sie die gewünschten Informationen aus.
- Die Option "Ergebnisdatei nach der Generierung anzeigen" ist für alle drei Ausgabeoptionen aktiv. Wenn diese Option aktiviert ist, wird die erzeugte Datei in der Browser-Ansicht (HTML-Ausgabe), MS Word (MS Word Ausgabe) und der Standardapplikation für .rtf-Dateien (RTF-Ausgabe) angezeigt.

 © 2007 Altova GmbH


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"/>

In der oben gezeigten Abbildung sehen Sie die generierte Schema-Dokumentation mit einem Index (alle verwandten Schemas mit ihren globalen Komponenten nach Komponententyp geordnet) am Beginn des Dokuments.

Anmerkung: Bei der Generierung von Dokumentation für W3C-Schemadokumente verwendet XMLSpy applikationsinterne Versionen dieser Dokumente. Aus diesem Grund werden andere Ordner für diese Applikationen nicht berücksichtigt und Redefinitionen und andere Änderungen am Schema werden in der Dokumentation nicht berücksichtigt.

15.6.4 Ansicht konfigurieren

Mit diesem Befehl können Sie die Anzeige des Content Model konfigurieren. Wenn Sie auf den Befehl klicken, wird rechts unten im XMLSpy-Fenster das Dialogfeld "Schema-Ansicht konfigurieren" geöffnet. Hier sehen Sie bereits bei der Eingabe in das Dialogfeld, wie sich Ihre Einstellungen auswirken. Sobald Sie auf OK klicken, werden die Einstellungen aktiv und für das Content Model aller XML-Schemadateien, die in der Folge geöffnet werden, übernommen. Diese Einstellungen werden auch für die erzeugte Schema-Dokumentation und die Druckausgabe übernommen.

Definieren von Eigenschaftsdeskriptorzeilen für das Content Model
Sie können festlegen, welche Element- und Attributeigenschaften in der Content Model-Ansicht angezeigt werden sollen. Diese Eigenschaften werden als Grid-Zeilen in Komponentenkästchen

angezeigt. Um Eigenschaftsdeskriptorzeilen zu definieren, gehen Sie vor wie folgt:

1. Wählen Sie **Schema-Design | Ansicht konfigurieren**. Daraufhin wird das Dialogfeld "Schema-Ansicht konfigurieren" angezeigt.
2. Klicken Sie auf dem Element- oder Attribut-Register auf das Symbol "Anhängen" oder "Einfügen" , um eine Eigenschaftsdeskriptorzeile hinzuzufügen. Die Zeile wird im Dialogfeld und zu den Elementkästchen in der Content Model-Ansicht hinzugefügt.
3. Wählen Sie aus der Dropdown-Liste die gewünschte Eigenschaft aus. *Siehe Abbildung unten.*
4. Wiederholen Sie Schritt 1 und 2 für alle benötigten Eigenschaften.

Die Content Model-Ansicht wird aktualisiert und zeigt die definierten Eigenschaftsdeskriptorzeilen für alle Elemente an, für die sie existieren.

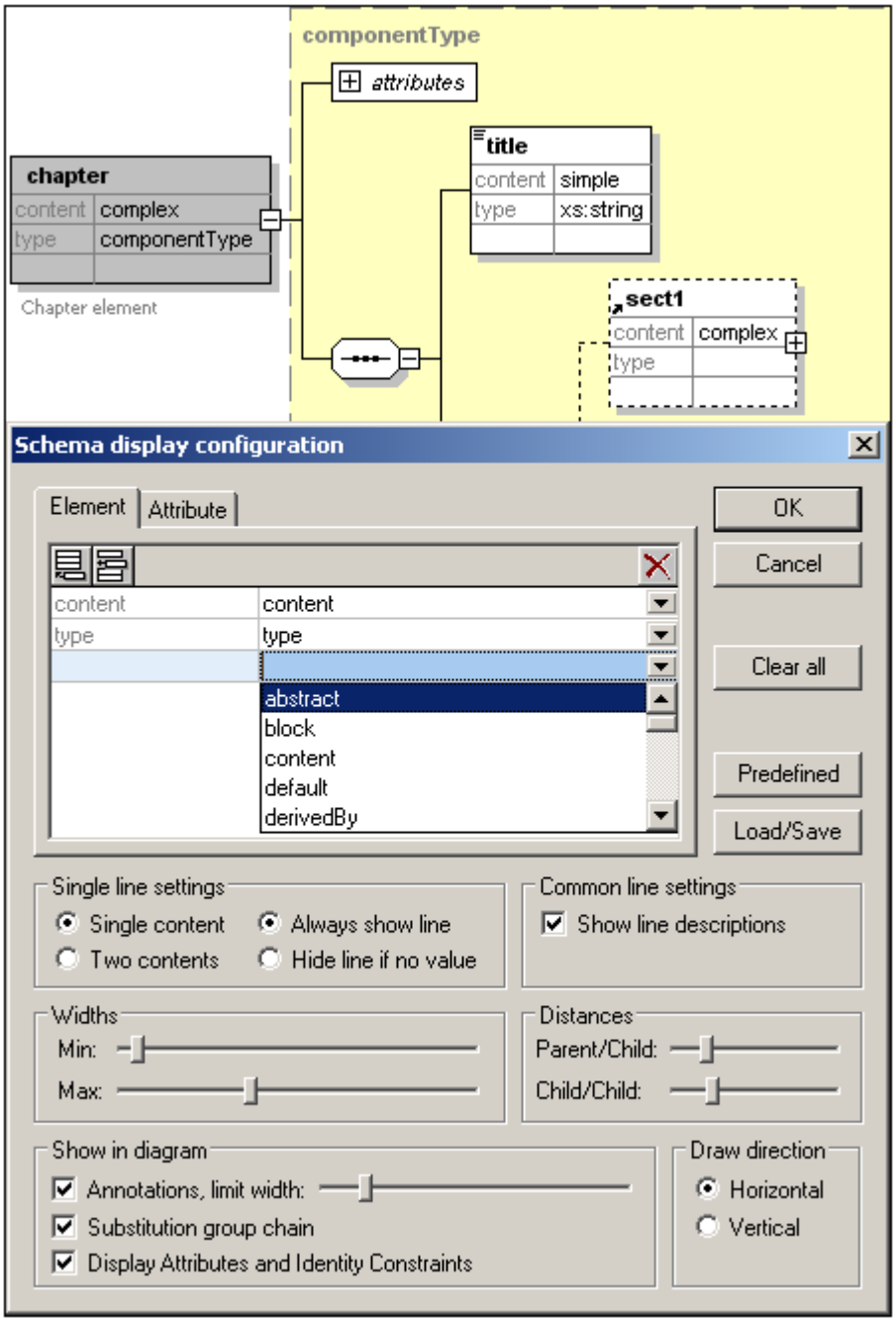
Bitte beachten Sie:

- Bei Attributen wird die von Ihnen definierte Konfiguration nur angezeigt, wenn Attribute (im Gegensatz zur Anzeige der Attribute in einem Bereich unterhalb der Content Model-Ansicht) im Diagramm angezeigt werden.
- Die konfigurierte Ansicht wird auf alle Content Model-Ansichten angewandt, die nach Definition der Ansicht, geöffnet werden.

Löschen einer Eigenschaftsdeskriptorzeile aus der Content Model-Ansicht
Um einzelne Eigenschaftsdeskriptorzeilen im Dialogfeld "Schema-Ansicht konfigurieren" zu löschen, wählen Sie die betreffende Zeile aus und klicken Sie auf das Symbol "Löschen" .

© 2007 Altova GmbH

Einstellungen zur Konfiguration der Content Model-Ansicht
Die Content Model-Ansicht kann mit Hilfe von Einstellungen im Dialogfeld " Schema-Ansicht konfigurieren" konfiguriert werden. Oben wurde beschrieben, wie Sie definieren, welche Eigenschaftsdeskriptorzeilen in der Content Model-Ansicht angezeigt werden. Im Folgenden werden die anderen Einstellungen beschrieben:



Zeileneinstellungen (gilt pro Zeile)
Sie können festlegen, ob eine Eigenschaftsdeskriptorzeile Einzel- oder Doppel-Inhalte enthalten soll und ob Zeilen für jedes einzelne Element oder nur für Elemente, die diese Eigenschaft enthalten, angezeigt werden sollen. Aktivieren Sie das entsprechende Optionsfeld für die gewünschte Einstellung. Beachten Sie, dass diese beiden Einstellungen separat für einzelne

Zeilen definiert werden können (markieren Sie die gewünschte Zeile und legen Sie die Einstellung fest).

Allgemeine Zeileneinstellungen
Mit dieser Option blenden Sie die Zeilenbeschreibungen (d.h. den Namen der Eigenschaft) ein bzw. aus.

Breite
Mit Hilfe dieser Schieberegler können Sie die Mindest- und Maximalgröße der Elementrechtecke in der Content Model-Ansicht einstellen. Ändern Sie die Größe, wenn der beschreibende Text nicht zur Gänze sichtbar ist oder wenn Sie die Anzeige standardisieren möchten.

Abstände
Mit den Schiebern können Sie den horizontalen und vertikalen Abstand zwischen den Elemente am Bildschirm verändern.

Im Diagramm zeigen
Das Kontrollkästchen Annotationen erlaubt es, den Annotationstext ein- oder auszublenden und die Textbreite des Annotationstextes zu verändern. Weiters können Sie die Anzeige der Substitutionsgruppen ein- oder ausblenden. Wenn Sie dieses Kontrollkästchen aktivieren, werden die Attribute und Identity Constraints im Content Model-Diagramm angezeigt; andernfalls werden Sie am unterern Rand des Content Model-Fensters als Register angezeigt.

Zeichenrichtung
Diese Optionen definieren die Orientierung des Elementdiagramms am Bildschirm. Wählen Sie zwischen horizontal und vertikal.

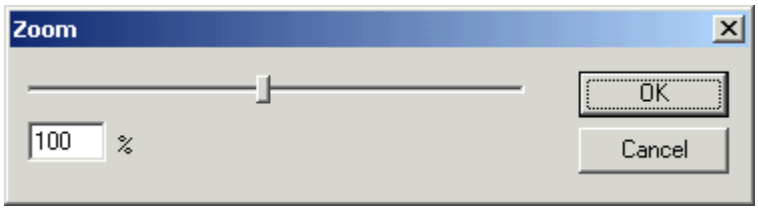
Editieren des Content Model im Diagramm selbst
Sie können Elementparameter- und Werte direkt im Inhaltsmodell ändern. Doppelklicken Sie auf die Eigenschaft, die Sie ändern wollen, und geben Sie einen neuen Wert ein. Wenn mehrere Werte zur Auswahl stehen, öffnet sich das Drop-down Listenfeld, ansonsten geben Sie den gewünschten Wert ein und bestätigen Sie mit der Eingabetaste.

Schaltflächen im Dialogfeld "Schema-Ansicht konfigurieren"

- Die Schaltfläche **Laden/Speichern** lädt und speichert die von Ihnen getätigten Einstellungen.
- Die Schaltfläche **Vordefiniert** setzt die Konfiguration der Ansicht auf die Werkseinstellungen zurück.
- Die Schaltfläche **Zurücksetzen** löscht alle Listenfeld-Einträge.

15.6.5 Vergrößern/Verkleinern

Mit diesem Befehl stellen Sie den Zoomfaktor der Ansicht des Content Model ein. Diese Funktion eignet sich zum Verkleinern eines großen Content Model, sodass das gesamte Content Model im Hauptfenster zu sehen ist. Der Zoomfaktor beträgt zwischen 10 % und 200 % der tatsächlichen Größe.

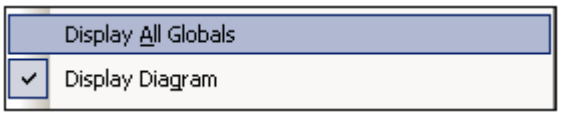


Ziehen Sie den Schieber um den Zoomfaktor zu verändern, oder geben Sie im Textfeld den

gewünschten Zoomfaktor ein.

15.6.6 Globale Elemente anzeigen

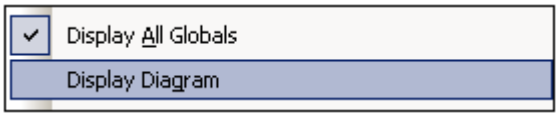
Mit diesem Befehl wechseln Sie von der Content Model-Ansicht zur Schemaübersicht, um alle globalen Komponenten im Schema anzuzeigen. Es ist entweder dieser Befehl oder der Befehl "Diagramm anzeigen" aktiv. Der jeweils aktive Befehl wird durch ein Häkchen auf der linken Seite angezeigt. (*siehe Abbildung unten).*



Alternativ dazu können Sie auch auf das Symbol **Globale Elemente anzeigen** am oberen Rand des Content Model klicken, um zur Schema-Übersicht zurückzukehren.

15.6.7 Diagramm anzeigen

Mit diesem Befehl zeigen Sie die Content Model-Ansicht der ausgewählten globalen Komponente an - falls die ausgewählte globale Komponente ein Content Model hat. Globale Komponenten mit einem Content Model (ComplexTypes, Elemente und Element-Gruppen) werden mit dem Symbol links davon gekennzeichnet. Es ist entweder dieser Befehl oder der Befehl "Globale Elemente anzeigen" aktiv. Der jeweils aktive Befehl wird durch ein Häkchen auf der linken Seite angezeigt. (*siehe Abbildung unten).*



Alternativ dazu haben Sie folgende Möglichkeiten, um zur Content Model-Ansicht zu wechseln:

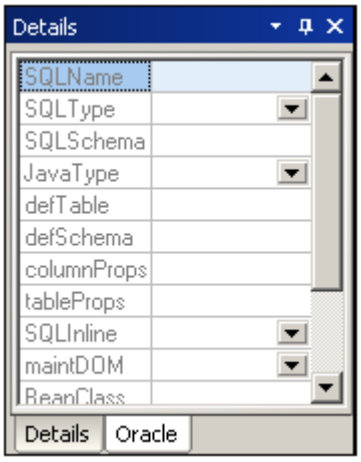
- Klicken Sie auf das Symbol neben der Komponente, die Sie anzeigen möchten, oder
- Doppelklicken Sie auf den Namen der Komponente im Komponenten-Navigator (oberste rechte Eingabehilfe).

15.6.8 Oracle Schema Extensions aktivieren

XMLSpy unterstützt Oracle Schema Extensions zur Verwendung in Oracle 9i Project XDB. Mit Hilfe dieser Extensions können Sie definieren und konfigurieren, wie Oracle 9i Project XDB XML-Dokumente intern speichert. Diese XML-Dokumente können dann über SQL Queries und Legacy Tools aufgerufen werden. Nähere Informationen dazu finden Sie auf der Oracle Website.

Bei Auswahl des Befehls **Oracle Schema Extensions aktivieren** geschieht Folgendes:

- Der XDB Namespace wird am `schema` Element deklariert: `xmlns:xdb="http://xmlns.oracle.com/xdb"`.
- In der Eingabehilfe "Details" wird ein Register "Oracle" angelegt, über das Sie Attribute - einschließlich XDB-spezifischer Attribute - zu Schema-Elementen wie `xsd:complexType` and `xsd:element` hinzufügen können.

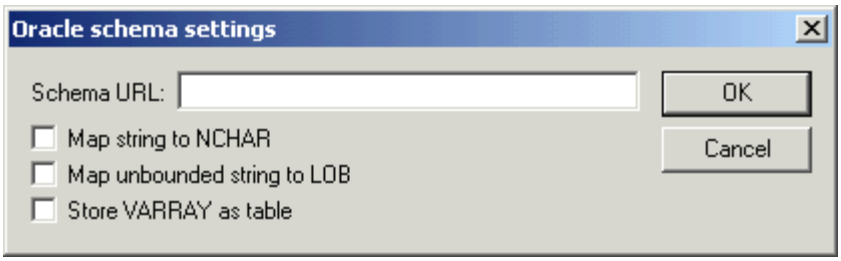


Oracle Extensions können für complexType, Elemente und Attribute definiert werden. Verwenden Sie die Eingabehilfe genau wie in XMLSpy.

Bitte beachten Sie: Sie können diesen Menübefehl aktivieren oder deaktivieren, d.h. Sie können Extensions ein- oder ausschalten. Bei aktivierten Oracle Extensions wird links neben dem Befehl ein Häkchen angezeigt. Bei Deaktivierung von Oracle Extensions (durch Klicken auf den aktivierten Befehl) werden die XDB Namespace Deklaration und alle XDB Extensions in der Datei gelöscht. Sie werden durch eine Meldung darauf hingewiesen, dass diese Informationen unwiderruflich gelöscht werden!

15.6.9 Oracle Schema-Einstellungen

Mit diesem Befehl können Sie globale Einstellungen für Oracle Schemaerweiterungen vornehmen.



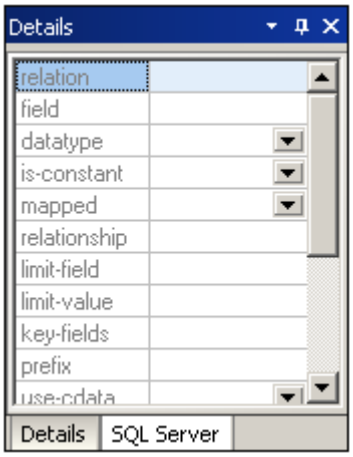
Diese Menüoption kann nur aufgerufen werden, wenn Sie zuvor Oracle Schema Extensions aktiviert haben (Menüoption "Oracle Schema Extensions aktivieren").

15.6.10 Microsoft SQL Server Schema Extensions aktivieren

XMLSpy unterstützt Microsoft SQL Server 2000 Schema Extensions zur Verwendung mit Microsoft SQL Server. Durch Verwendung dieser Extensions können Sie definieren und konfigurieren, wie Microsoft SQL Server Ihre XML-Dokumente intern speichert. Diese XML-Dokumente können dann über SQL-Abfragen und Legacy Tools aufgerufen werden. Weitere Informationen zu diesem Thema finden Sie auf der Microsoft Website.

Bei Auswahl dieser Menüoption geschieht Folgendes:

- Der SQL Server Namespace wird am `schema` Element deklariert: `xmlns:sql="urn:schemas-microsoft-com:mapping-schema"`.
- In der Eingabehilfe "Details" wird ein Register "SQL Server" angelegt, über das Sie Attribute zu Schema-Elementen wie `xsd:element` hinzufügen können.



Wenn SQL Server Extensions für eine Schema-Komponente definiert werden können, steht das Register "SQL Server" in der Eingabehilfe "Details" zur Verfügung, wenn die Komponente ausgewählt ist. Verwenden Sie die Eingabehilfe genau wie in XMLSpy.

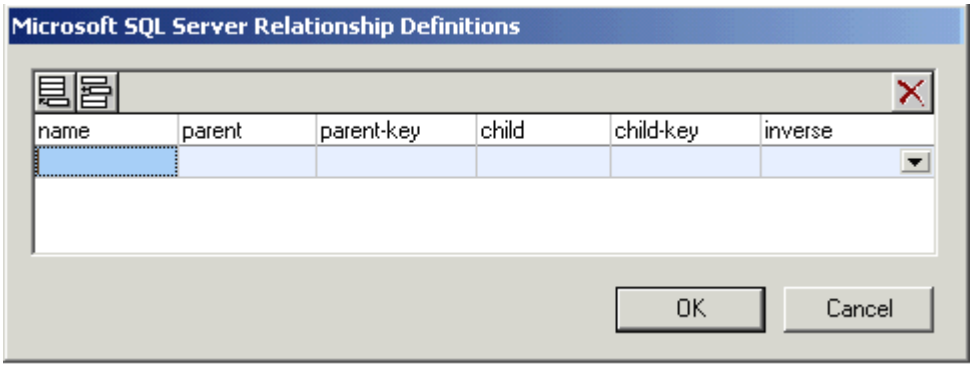
Bitte beachten Sie: Sie können diesen Menübefehl aktivieren oder deaktivieren, d.h. Sie können Extensions ein- oder ausschalten. Bei aktivierten SQL Server Extensions wird links neben dem Befehl ein Häkchen angezeigt. Bei Deaktivierung von SQL Server Extensions (durch Klicken auf den aktivierten Befehl) werden die SQL Server Namespace Deklaration und alle SQL Extensions in der Datei gelöscht. Sie werden durch eine Meldung darauf hingewiesen, dass diese Informationen unwiderruflich gelöscht werden!

15.6.11 Named Schema Relationships

Mit diesem Befehl können Sie "named relationships" definieren, die benötigt werden, um das Erstellen von Dokumenthierarchien zu erlauben.
Diese Menüoption kann nur aufgerufen werden, wenn Sie zuvor SQL Server Schema Extensions aktiviert haben (Menüoption "SQL Server Schema Extensions aktivieren").

So erstellen Sie Named Schema Relationships:

1. Klicken Sie auf das Symbol "Einfügen" oder "Anhängen" , um im Dialogfeld eine neue Zeile einzufügen.
2. Klicken Sie in das neue Feld, um den Namen der Beziehung einzugeben.
3. Mit OK bestätigen Sie Ihre Eingaben.



Es wird ein SQL Relationship Element erzeugt, das unter der Namespace Deklaration eingefügt

wird.

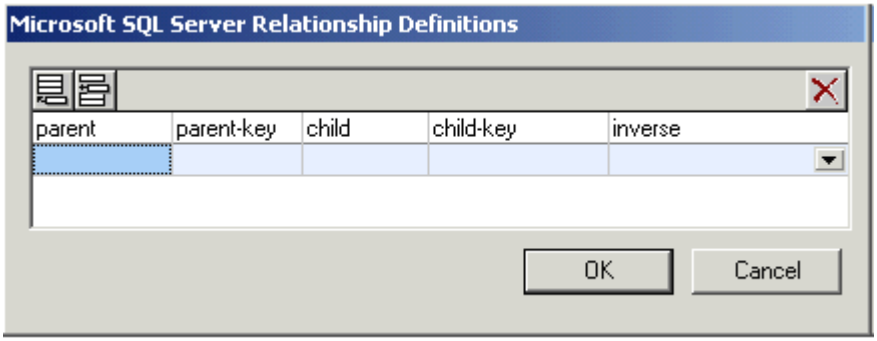
Bitte beachten Sie: Klicken Sie auf das Symbol "Löschen" um eine Zeile aus dem Dialogfeld zu löschen.

15.6.12 Unnamed Element Relationships

Mit diesem Befehl können Sie Unnamed Relationships definieren, um Informationen für das Erstellen von Dokumenthierarchien bereitstellen zu können. Diese Menüoption kann nur aufgerufen werden, wenn Sie zuvor SQL Server Schema Extensions aktiviert haben (Menüoption **Microsoft SQL Server Schema Extensions aktivieren**).

So erstellen Sie Unnamed Schema Relationships:

1. Klicken Sie auf das Symbol "Einfügen" oder "Anhängen", um im Dialogfeld eine neue Zeile einzufügen.
2. Klicken Sie in das Feld, um den Namen der Beziehung einzugeben.
3. Mit OK bestätigen Sie Ihre Eingaben.



Hierdurch wird ein SQL Relationship Element für das ausgewählte Schemaelement erzeugt.

Bitte beachten Sie: Klicken Sie auf das Symbol "Löschen" um eine Zeile aus dem Dialogfeld zu löschen.

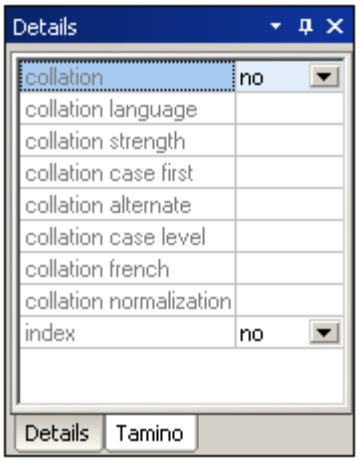
15.6.13 Tamino Schema Extensions aktivieren

XMLSpy unterstützt Tamino Schema Extensions zur Verwendung mit Tamino Server. Software AG und die Altova GmbH haben ein weltweit gültiges Abkommen unterzeichnet, mit dem Benutzer von XMLSpy einfachen und kostengünstigen Zugang zu Tamino (über ein Tamino Datenbankbündel) erhalten. Weitere Informationen erhalten Sie im Abschnitt Tamino. Durch Verwendung der Tamino Schema Extensions können Sie definieren, wie Tamino Ihre XML-Dokumente intern speichert. Diese XML-Dokumente können dann über Abfragetools und Legacy Tools aufgerufen werden.

Bei Auswahl dieser Menüoption geschieht Folgendes:

- Wenn das XML-Schema Namespace Präfix nicht `xs:` ist, werden Sie von XMLSpy gefragt, ob das Präfix in das von Tamino benötigte Präfix in `xs:` geändert werden darf.
- Wenn die Dateierweiterung nicht `.tsd` ist, werden Sie von XMLSpy gefragt, ob die Dateierweiterung in die von Tamino benötigte Erweiterung `.tsd`. geändert werden darf.
- Es erscheint das Dialogfeld "Tamino-Eigenschaften", in dem Sie die Tamino-Eigenschaften einstellen können. Nähere Informationen dazu siehe Tamino-Eigenschaften.

- Der Tamino Namespace wird am `schema` Element: `xmlns:tsd="http://namespaces.softwareag.com/tamino/TaminoSchemaDefinition"` deklariert.
- In der Eingabehilfe "Details" wird ein Register "Tamino" angelegt, über das Sie Attribute zu Schema-Elementen wie `xsd:element` hinzufügen können.



Wenn Tamino Extensions für eine Schema-Komponente definiert werden können, steht das Register "Tamino" in der Eingabelhilfe "Details" zur Verfügung, wenn die Komponente ausgewählt ist. Verwenden Sie die Eingabehilfe genau wie in XMLSpy.

Bitte beachten Sie: Sie können diesen Menübefehl aktivieren oder deaktivieren, d.h. Sie können Extensions ein- oder ausschalten. Bei aktivierten Tamino Extensions wird links neben dem Befehl ein Häkchen angezeigt. Bei Deaktivierung von Tamino Extensions (durch Klicken auf den aktivierten Befehl) werden die Tamino Namespace Deklaration und alle Tamino Extensions in der Datei gelöscht. Sie werden durch eine Meldung darauf hingewiesen, dass diese Informationen unwiderruflich gelöscht werden!

15.6.14 Tamino Schema-Eigenschaften

Mit diesem Befehl öffnen Sie das Dialogfeld "Tamino-Eigenschaften". Hier erhalten Sie Informationen zur Tamino Datenbank und Schemaeigenschaften. Nähere Informationen zu diesen Eigenschaften finden Sie im Abschnitt Tamino dieser Dokumentation unter Tamino Schema-Eigenschaften.

15.6.15 Verbindung zu SchemaAgent Server herstellen

Dieser Befehl ist aktiv, wenn ein XML-Schema-Dokument aktiv ist. Sie können damit eine Verbindung zu einem SchemaAgent Server herstellen. Dies ist jedoch nur möglich, wenn auf Ihrem Rechner ein lizenzierter Altova SchemaAgent 2007 installiert ist. Wenn Sie auf diesen Befehl klicken, wird das Dialogfeld "Verbindung zu SchemaAgent herstellen" (*siehe Screenshot unten*) angezeigt:

Sie können entweder mit dem lokalen Server arbeiten (dem SchemaAgent, Server, der mit Altova SchemaAgent 2007 mitgeliefert wird). oder mit einem Netzwerk-Server (dem Altova SchemaAgent Server 2007, der kostenlos erhältlich ist). Wenn Sie die Option "Lokal arbeiten" auswählen, wird der lokale Server von SchemaAgent gestartet, wenn Sie auf OK klicken und es wird eine Verbindung mit diesem Server hergestellt. Wenn Sie die Option "Verbindung zu Netzwerk-Server herstellen" wählen, muss der ausgewählte SchemaAgent Server ausgeführt werden, damit eine Verbindung hergestellt werden kann.

Wenn Sie eine Verbindung zu SchemaAgent Server hergestellt haben, fungiert XMLSpy als SchemaAgent Client. Sie haben darin leistungsstarke Funktionen zum Editieren und Verwalten von Schemas zur Verfügung. Nähere Informationen zu SchemaAgent, zur Installation von SchemaAgent Server und zum Herstellen einer Verbindung mit SchemaAgent Server finden Sie im Abschnitt Schema/WSDL-Ansicht dieses Benutzerhandbuchs unter Arbeiten mit SchemaAgent. Nähere Informationen über die Installation der beiden Produkte und die Verwendung finden Sie im SchemaAgent-Benutzerhandbuch zu diesen Produkten.

Nachdem Sie eine Verbindung mit SchemaAgent Server hergestellt haben, erscheint in der Leiste am oberen Rand des Hauptfensters eine Meldung mit Informationen über die Verbindung. Sie haben nun uneingeschränkten Zugriff auf alle Schemas und Schema-Komponenten im/in den Suchpfad(en) (Ordner(n)), die für den SchemaAgent Server definiert wurden, mit dem XMLSpy verbunden ist.

Bitte beachten Sie: Damit eine Verbindung zustande kommt, müssen Sie den Altova SchemaAgent Client auf Ihrem Rechner installiert haben und eine gültige Lizenz dafür besitzen.

15.6.16 Verbindung mit SchemaAgent Server trennen

Dieser Befehl steht zur Verfügung, wenn eine Verbindung zu einem SchemaAgent Server zustandegekommen ist. Bei Auswahl dieses Befehls wird die Verbindung zwischen XMLSpy und dem SchemaAgent Server getrennt.

15.6.17 In SchemaAgent anzeigen

Mit diesem Befehl werden das aktive Schema und optional damit verknüpfte Schemas im Altova-Produkt SchemaAgent 2007 angezeigt. (Dieses Produkt muss auf demselben Rechner wie XMLSpy installiert sein, wenn Sie die Funktionen von SchemaAgent verwenden möchten). Das/die Schema(s) wird/werden in einem neuen SchemaAgent Design in SchemaAgent geöffnet.

© 2007 Altova GmbH

Wenn Sie den Mauszeiger über den Menübefehl **In SchemaAgent anzeigen** halten, erscheint ein Untermenü, das Optionen enthält, welche Schemas in SchemaAgent angezeigt werden sollen. Nähere Informationen zu diesen Optionen finden Sie in dieser Dokumentation im Abschnitt "Schema/WSDL-Ansicht" unter Anzeigen von Schemas in SchemaAgent.

15.6.18 Erweiterte Validierung

Mit diesem Befehl können Sie das aktuell aktive Schema und die damit verwandten Schemas validieren. Eine Beschreibung zu dieser Funktion finden Sie in dieser Dokumentation im Abschnitt "Schema/WSDL-Ansicht" unter Erweiterte Validierung.

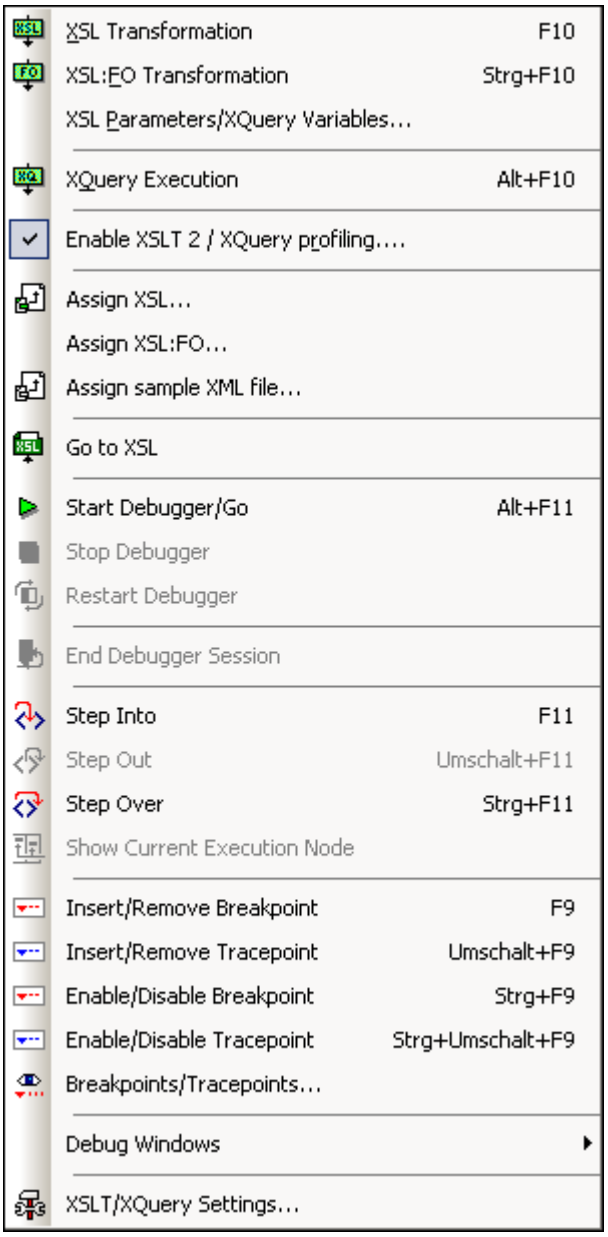
15.7 Menü "XSL/XQuery"

Mit Hilfe der XSL Transformationssprache können Sie festlegen, wie ein XML-Dokument in andere XML-Dokumente oder Textdateien konvertiert werden soll. Eine Art von XML-Dokument, die mittels eines XSLT-Dokuments generiert wird, ist ein FO-Dokument, das anschließend weiter in ein PDF-Ausgabedokument konvertiert wird.
XMLSpy enthält integrierte XSLT-Prozessoren (für XSLT 1.0 und XSLT 2.0) und kann mit einem FO-Prozessor auf Ihrem System verknüpft werden, um XML-Dateien zu transformieren und verschiedene Arten von Ausgabedokumenten zu generieren. Der Pfad zum FO-Prozessor muss im Dialogfeld "Optionen" auf dem Register XSL (Extras | Optionen) angegeben werden, um direkt innerhalb der XMLSpy Benutzeroberfläche verwendet werden zu können.

Des Weiteren verfügt XMLSpy über einen integrierten XQuery-Prozessor, mit Hilfe dessen Sie XQuery-Dokumente (mit oder ohne Referenz auf ein XML-Dokument) ausführen können.

Die Befehle für die oben beschriebenen Transformationen finden Sie im Menü **XSL/XQuery**. Zusätzlich dazu enthält dieses Menü auch Befehle für die Arbeit mit dem Altova XSLT/XQuery Debugger.

© 2007 Altova GmbH



15.7.1 XSL-Transformation

F10

Mit diesem Befehl führen Sie mittels eines zugewiesenen XML-Dokuments eine **XSL-Transformation** durch. Die Transformationen kann entweder mit Hilfe des entsprechenden Altova XSLT-Prozessors (Altova XSLT 1.0-Prozessor für XSLT 1.0 Stylesheets; Altova XSLT 2.0-Prozessor für XSLT 2.0 Stylesheets), mit Hilfe des MSXML-Moduls von Microsoft oder eines externen XSLT-Prozessors durchgeführt werden. Der in Verbindung mit diesem Befehl verwendete Prozessor, wird im Dialogfeld "Optionen" (**Extras | Optionen**) auf dem Register XSL festgelegt.

Sollte Ihr XML-Dokument eine Referenz zu einem XSLT Stylesheet enthalten, wird dieses

Stylesheet automatisch für die XSLT Transformation verwendet. (Ein XSLT Stylesheet kann über den Befehl XSL zuweisen zugewiesen werden), Wenn das XML-Dokument Teil eines Projekts ist, können Sie im Dialogfeld "Projekteigenschaften" ein XSLT Stylesheet auch für einzelne Ordner definieren. Klicken Sie mit der rechten Maustaste auf den/die Projektordner oder die zu transformierenden Datei(en) und wählen Sie XSL-Transformation. Wurde einer XML-Datei kein XSLT Stylesheet zugewiesen, werden Sie aufgefordert das XSLT Stylesheet anzugeben.

Automatisieren von XSLT-Transformationen mit Altova XML 2007
Altova XML ist eine kostenlose Applikation, die den Altova XML-Validator sowie einen XLST 1.0-, XSLT 2.0- und XQuery 1.0-Prozessor enthält. Das Tool kann über die Befehlszeile, über eine COM-Schnittstelle, in Java-Programmen und in .NET-Applikationen zur Validierung von XML-Dokumenten, zum Transformieren von XML-Dokumenten unter Verwendung von XSLT 1.0 und 2.0 Stylesheets sowie zum Ausführen von XQuery-Dokumenten verwendet werden.

XSLT-Transformationen können daher mittels Altova XML automatisiert werden. So können Sie z.B. eine Batch-Datei erstellen, die Altova XML aufruft, um damit eine Reihe von XML-Dokumenten zu transformieren. Nähere Informationen dazu finden Sie in der Altova XML Dokumentation.

15.7.2 XSL:FO-Transformation

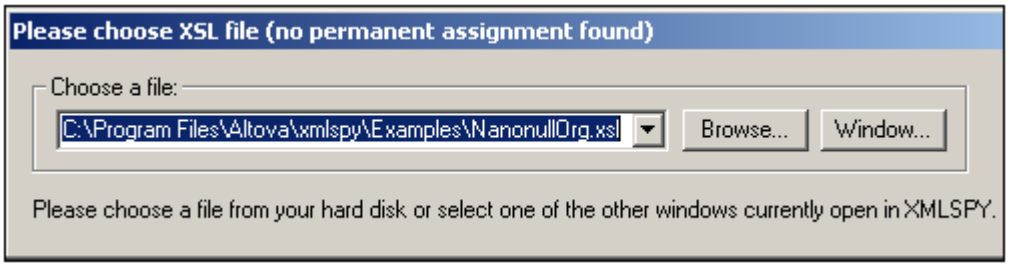
Strg+F10

FO ist ein XML-Format zur Beschreibung von seitenweise formatierten Dokumenten. Ein FO-Prozessor wie z.B. der FOP-Prozessor von Apache XML Projects verarbeitet FO, um eine PDF-Datei zu erzeugen, die Generierung eines PDF-Dokuments erfolgt also in zwei Schritten.

1. Das XML-Dokument wird mittels eines XSLT Stylesheet (auch als XSL-FO bezeichnet) in ein FO-Dokument transformiert.
2. Aus dem FO-Dokument wird ein PDF-Dokument (oder ein anderes Ausgabeformat) generiert.

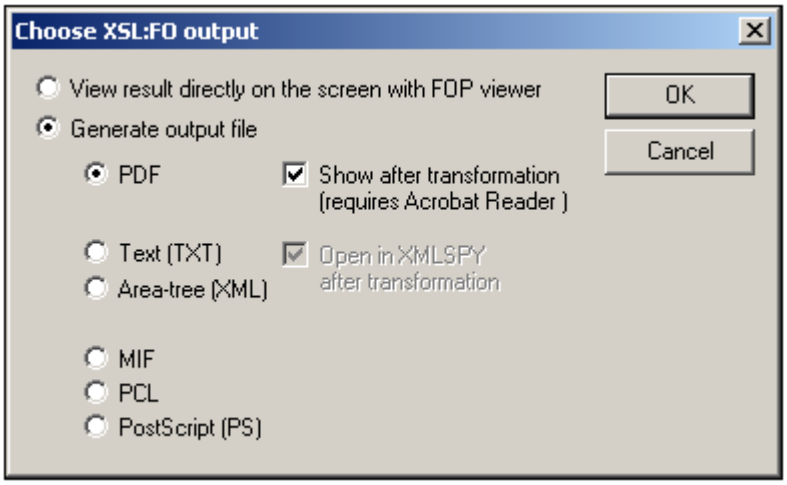
Mit dem Befehl **XSL:FO-Transformation** wird ein XML-Dokument oder ein FO-Dokument ins PDF-Format transformiert.

- Wird die **XSL:FO-Transformation** an einem XML-Quelldokument durchgeführt, werden beide oben beschriebenen Schritte nacheinander durchgeführt. Wenn das für die Transformation in FO benötigte XSLT (oder XSL-FO) Stylesheet im XML-Dokument nicht referenziert ist, werden Sie aufgefordert, eines für die Transformation zuzuweisen (*siehe Abbildung unten*).



Das erzeugte FO-Dokument wird von dem FO-Prozessor, der im Dialogfeld "Optionen" (**Extras | Optionen**), Register XSL definiert wurde, direkt verarbeitet.
Wird der Befehl **XSL:FO-Transformation** an einem FO-Dokument ausgeführt, wird das Dokument mit dem FO-Prozessor, der im Dialogfeld "Optionen" (**Extras | Optionen**), Register XSL definiert wurde, verarbeitet.

XSL:FO-Transformationsausgabe
Wenn Sie den Befehl **XSL:FO-Transformation** aufrufen, wird das Dialogfeld "XSL:FO-Ausgabe wählen" angezeigt (*siehe Abbildung unten*). (Wenn dem aktiven XML-Dokument kein XSLT Stylesheet zugewiesen wurde, werden Sie zuerst nach der XSLT-Datei gefragt).



Sie können das vom FO-Prozessor erzeugte Dokument mit Hilfe des FOP-Viewers direkt am Bildschirm anzeigen oder Sie können eine Ausgabedatei in einem der folgenden Formate erzeugen: PDF, Text, als XML-Baumdiagramm, MIF PCL oder PostScript.

Bitte beachten Sie:

- Der Apache FOP-Prozessor kann kostenlos über den Link im Altova Download Center heruntergeladen werden. Laden Sie FOP herunter und installieren Sie den Prozessor und geben Sie anschließend den Pfad zur FOP Batch-Datei im Dialogfeld "Optionen" (**Extras | Optionen**) auf dem Register XSL an.
- Sie können den Befehl XSL:FO-Transformation nicht nur an der aktiven Datei im Hauptfenster anwenden, sondern auch an jeder Datei/jedem beliebigen Ordner die/den Sie im Projektordner auswählen. Klicken Sie dazu mit der rechten Maustaste und wählen Sie den Befehl **XSL:FO-Transformation**. Es wird das XSLT Stylesheet verwendet, das dem ausgewählten Projektordner zugewiesen wurde.

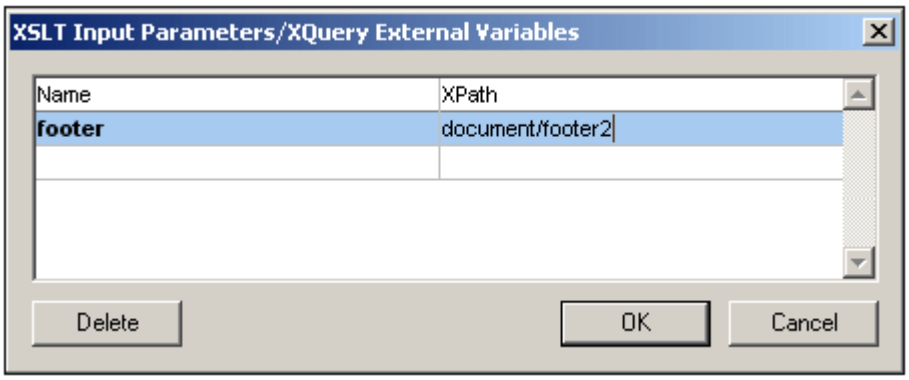
15.7.3 XSL-Parameter/XQuery-Variablen

Mit dem Befehl **XSL-Parameter/XQuery-Variablen** öffnen Sie das Dialogfeld "XSLT-Eingabeparameter/externe XQuey-Variablen" (*siehe Screenshot*). Sie können den Namen und die dazugehörigen Werte von einem oder mehreren Parametern eingeben, die Sie an das XSLT Stylesheet übergeben möchten, oder den Namen und die dazugehörigen Werte von einer oder mehreren XQuery-Variablen, die an das XQuery-Dokument übergeben werden sollen. Der/die Parameterwert(e) werden an den/die Parameter in der ausgewählten XSLT-Datei übergeben, wenn eine XML-Datei mit Hilfe des Befehls **XSL-Transformation** aus dem XSL-Menü verarbeitet wird. Wenn Sie den Befehl **XQuery-Ausführung** auswählen, werden externe XQuery-Variablenwerte an das XQuery-Dokument übergeben.

Bitte beachten Sie: Parameter oder Variablen, die Sie im Dialogfeld "XSLT-Eingabeparameter/externe XQuery-Variablen" eingeben, werden nur an den integrierten Altova XSLT-Prozessor übergeben. Wenn Sie daher MSXML oder einen anderen externen Prozessor verwenden, den Sie konfiguriert haben, werden diese Parameter nicht an diesen

Prozessor übergeben.

Verwendung von XSLT-Parametern
Bei dem Wert, den Sie für den Parameter eingeben, kann es sich um einen XPath ohne Anführungszeichen oder einen von Anführungszeichen eingeschlossenen Textstring handeln.



Bitte beachten Sie: Sobald Sie im Dialogfeld "XSLT-Eingabeparameter/externe XQuery-Variablen" die Parameterwerte eingegeben haben, werden diese für alle zukünftigen Transformationen verwendet, bis sie ausdrücklich gelöscht werden oder die Applikation neu gestartet wird. Parameter, die im Dialogfeld "XSLT-Eingabeparameter/externe XQuery-Variablen" eingegeben wurden, werden auf Applikationsebene definiert und werden von diesem Punkt an bei jeder Transformation, die über die IDE durchgeführt wird, an das jeweilige XSLT-Dokument übergeben. Dies bedeutet:

- Parameter sind nicht mit einem bestimmten Dokument verknüpft und
- Alle Parameter, die im Dialogfeld "XSLT-Eingabeparameter/externe XQuery-Variablen" eingegeben wurden, werden gelöscht, sobald die Applikation (XMLSpy) geschlossen wird.

Verwendung von XSLT-Parametern
Im folgenden Beispiel wählen wir die für das PDF-Dokument benötigte Fußzeile aus drei Wahlmöglichkeiten im XML-Dokument (`footer1`, `footer2`, `footer3`) aus.

```
<?xml version="1.0" encoding="UTF-8"?>
<document xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:noNamespaceSchemaLocation="C:\workarea\footers\footers.xsd">
   <footer1>Footer 1</footer1>
   <footer2>Footer 2</footer2>
   <footer3>Footer 3</footer3>
   <title>Document Title</title>
   <para>Paragraph text.</para>
   <para>Paragraph text.</para>
</document>
```

Die XSLT-Datei enthält in der Vorlage für das Root-Element einen lokalen Parameter namens `footer`. Dieser Parameter hat einen Standardwert `footer1`. Der Parameter wird anschließend in der Vorlage in der Definition des Footer-Blocks mit einem Wert `$footer` instanziiert.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
   xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
   xmlns:fo="http://www.w3.org/1999/XSL/Format">
   ...
```

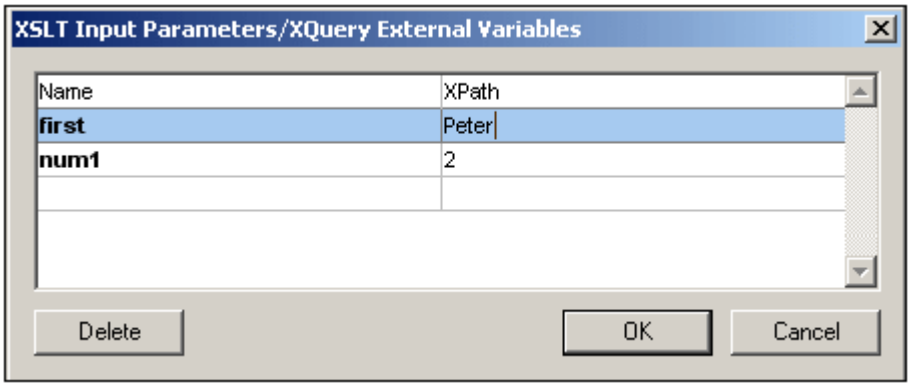
```
<xsl:template match="/">
   <xsl:param name="footer" select="document/footer1" />
   <fo:root>
      <xsl:copy-of select="$fo:layout-master-set" />
      <fo:page-sequence master-reference="default-page"
         initial-page-number="1" format="1">
         <fo:static-content flow-name="xsl-region-after"
            display-align="after">
            ...
            <fo:inline color="#800000" font-size="10pt" font-weight="bold">
               <xsl:value-of select="$footer"/>
            </fo:inline>
            ...
         </fo:static-content>
      </fo:page-sequence>
   </fo:root>
</xsl:template>
</xsl:stylesheet>
```

Sie können im XSLT-Eingabeparameter-Dialogfeld einen neuen Wert für den Parameter `footer` eingeben, z.B. den XPath: `document/footer2` (*siehe Abbildung oben)* oder einen Textstring. Während der Transformation wird dieser Wert an den Parameter `footer` in der Vorlage für das Root-Element übergeben und beim Instanziieren des Footer-Blocks verwendet.

Bitte beachten Sie:

- Wenn Sie den Befehl **XSL:FO-Transformation** (**XSL/XQuery | XSL:FO-Transformation**) verwenden, werden Parameter, die im Dialogfeld "XSLT-Eingabeparameter/externe XQuery-Variablen" eingegeben werden, **nicht** an das Stylesheet übergeben. Damit diese Parameter in der PDF-Ausgabe verwendet werden, führen Sie mittels des Befehls XSLT-Transformation **(XSL/XQuery | XSL-Transformation)** zuerst eine XSL-FO-Transformation durch und anschließend mit Hilfe des Befehls **XSL:FO-Transformation** (**XSL/XQuery | XSL:FO-Transformation**) eine FO-PDF-Transformation.
- Falls Sie einen anderen als den integrierten Altova XSLT-Prozessor verwenden, werden Parameter, die Sie mit Hilfe des Befehls "XSLT-Eingabeparameter" eingeben, nicht an den externen Prozessor übergeben.

Verwendung von externen XQuery-Variablen
Beim Wert, den Sie für eine externe XQuery-Variable eingeben, muss es sich um einen Literalwert ohne Anführungszeichen handeln. Der Datentyp der externen Variable wird in der Variablendeklaration im XQuery-Dokument angegeben.



Bitte beachten Sie: Sobald Sie im Dialogfeld "XSLT-Eingabeparameter/externe XQuery-Variablen" eine Reihe von externen XQuery-Variablen eingegeben haben, werden diese für alle zukünftigen Transformationen verwendet, bis sie ausdrücklich gelöscht werden

oder die Applikation neu gestartet wird. Variablen, die im Dialogfeld "XSLT-Eingabeparameter/externe XQuery-Variablen" eingegeben wurden, werden auf Applikationsebene definiert und werden von diesem Punkt an bei jeder Transformation, die über die IDE durchgeführt wird, an das jeweilige XSLT-Dokument übergeben. Dies bedeutet:

- Variablen sind nicht mit einem bestimmten Dokument verknüpft und
- Alle Variablen, die im Dialogfeld "XSLT-Eingabeparameter/externe XQuery-Variablen" eingegeben wurden, werden gelöscht, sobald die Applikation (XMLSpy) geschlossen wird.

Anwendungsbeispiel für externe XQuery-Variablen
Im folgenden Beispiel wird im XQuery-Dokument eine Variable `$first` deklariert und anschließend in der Return Clause des FLWOR-Ausdrucks verwendet:

```
xquery version "1.0";
declare variable $first as xs:string external;
let $last := "Jones"
return concat( $first, " ", $last )
```

Diese XQuery gibt `Peter Jones` zurück, wenn der Wert der externen Variable (der im Dialogfeld "XSLT-Eingabeparameter/externe XQuery-Variablen" eingegeben wurde) `Peter` ist. Beachten Sie Folgendes:

- Das externe Schlüsselwort in der Variablendeklaration im XQuery-Dokument gibt an, dass es sich bei dieser Variable um eine externe Variable handelt.
- Die Definition des statischen Typs der Variable ist optional. Wenn der Datentyp für die Variable nicht in der Variablendeklaration angegeben wurde, wird dem Variablenwert der Typ `xs:untypedAtomic` zugewiesen.
- Wenn im XQuery-Dokument eine externe Variable deklariert wurde, aber keine externe Variable dieses Namens an das XQuery-Dokument übergeben wird, wird eine Fehlermeldung ausgegeben.
- Wenn eine externe Variable deklariert und im Dialogfeld "XSLT-Eingabeparameter/externe XQuery-Variablen" eingegeben wurde, wird diese als Gültigkeitsbereich für das auszuführende XQuery-Dokument verwendet. Wird eine neue Variable dieses Namens im XQuery-Dokument deklariert, setzt diese die externe Variable für den Gültigkeitsbereich vorübergehend außer Kraft. Das unten gezeigte XQuery-Dokument gibt z.B. `Paul Jones` zurück, obwohl die externe in-scope Variable `$first` den Wert `Peter` hat.

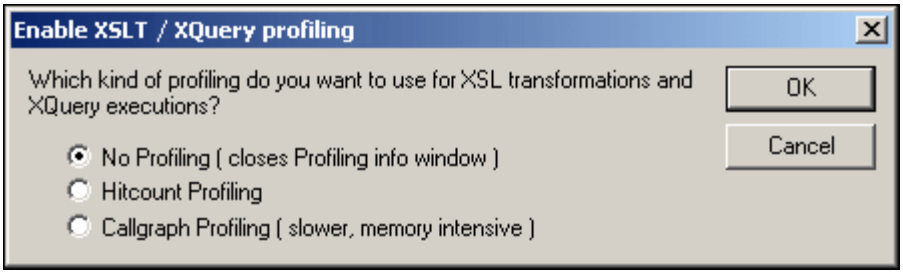
  ```
  xquery version "1.0";
  declare variable $first as xs:string external;
  let $first := "Paul"
  let $last := "Jones"
  return concat( $first, " ", $last )
  ```

Bitte beachten Sie: Es handelt sich nicht um einen Fehler, wenn im Dialogfeld "XSLT-Eingabeparameter/externe XQuery-Variablen" eine externe XQuery-Variable (oder ein XSLT-Parameter) definiert ist, diese im XQuery-Dokument aber nicht verwendet wird. Auch wenn ein XSLT-Parameter (oder eine externe XQuery-Variable) im Dialogfeld "XSLT-Eingabeparameter/externe XQuery-Variablen" definiert ist, aber in einer XSLT-Transformation nicht verwendet wird, handelt es sich nicht um einen Fehler.

© 2007 Altova GmbH

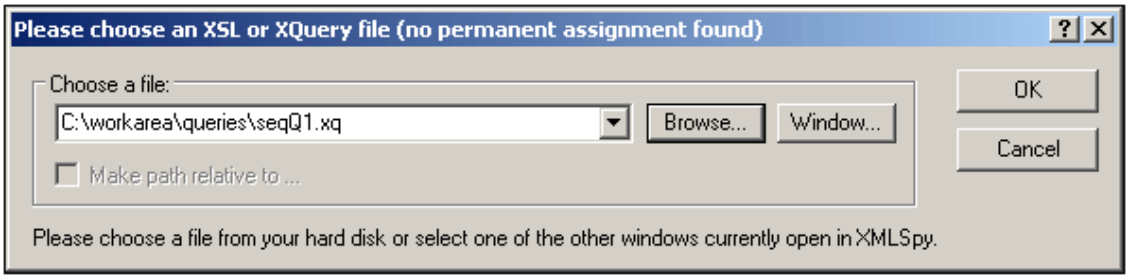
15.7.4 XSLT 2 / XQuery-Profilerstellung aktivieren

Mit dem Befehl **XSLT 2 / XQuery-Profilerstellung aktivieren** öffnen Sie das gleichnamige Dialogfeld. In diesem Dialogfeld können Sie den Profiler aktivieren, ein Tool, das analysiert, wie lange die Ausführung von Anweisungen während einer XSLT-Transformation/XQuery-Ausführung dauert.



15.7.5 XQuery-Ausführung

Mit diesem Befehl wird ein XQuery-Dokument ausgeführt. Der Befehl steht zur Verfügung, wenn eine XQuery- oder XML-Datei aktiv ist. Bei Aufruf von einer XML-Datei aus erscheint ein Dialogfeld, in dem Sie aufgefordert werden, die XML-Datei mit einer XQuery-Datei zu verknüpfen.



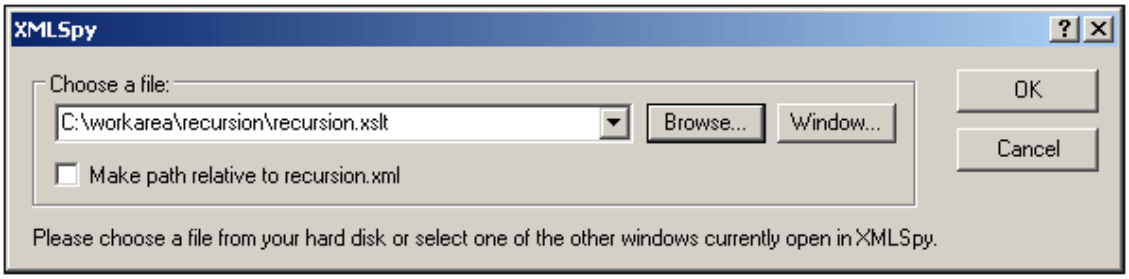
Automatisieren der XQuery-Ausführung mit Altova XML 2007

Altova XML ist eine kostenlose Applikation, die den Altova XML-Validator sowie einen XLST 1.0-, XSLT 2.0- und XQuery 1.0-Prozessor enthält. Das Tool kann über die Befehlszeile, über eine COM-Schnittstelle, in Java-Programmen und in .NET-Applikationen zur Validierung von XML-Dokumenten, zum Transformieren von XML-Dokumenten unter Verwendung von XSLT 1.0 und 2.0 Stylesheets sowie zum Ausführen von XQuery-Dokumenten verwendet werden.

XQuery-Aufgaben können daher mittels Altova XML automatisiert werden. So können Sie z.B. eine Batch-Datei erstellen, die Altova XML aufruft, um eine Reihe von XQuery-Dokumenten auszuführen. Nähere Informationen dazu finden Sie in der Altova XML Dokumentation.

15.7.6 XSL zuweisen...

Mit diesem Befehl weisen Sie einem XML-Dokument ein XSLT Stylesheet zu. Wenn Sie auf den Befehl klicken, wird ein Dialogfeld geöffnet, in dem Sie angeben können, welche XSLT-Datei zugewiesen werden soll.



Es wird eine `xml-stylesheet` Verarbeitungsanweisung in das XML-Dokument eingefügt:

```
<?xml-stylesheet type="text/xsl"
   href="C:\workarea\recursion\recursion.xslt"?>
```

Beachten Sie, dass Sie den Pfad zur zugewiesenen Datei durch Aktivieren des Kontrollkästchens "Pfad relativ machen zu..." relativ machen können.

15.7.7 XSL:FO zuweisen...

Mit diesem Befehl weisen Sie einem XML-Dokument ein XSL Stylesheet für die FO-Transformation zu. Der Befehl öffnet ein Dialogfeld, in dem Sie angeben können, welche XSL oder XSLT Datei zugewiesen werden soll. Die benötigte Verarbeitungsanweisung wird in das XML-Dokument eingefügt:

```
<?xmlspyxslfo C:\Program Files\Altova\xmlspy\Examples\OrgChartFO.xsl?>
```

Durch Aktivieren des Kontrollkästchens "Pfad relativ machen zu..." können Sie den Pfad der zugewiesenen Datei relativ machen. Ein XML-Dokument kann zwei zugewiesene XSLT-Dateien haben: Eine Standard XSLT-Transformationsdatei, sowie eine zweite Datei für eine XSLT-Transformation in FO.

15.7.8 XML-Beispieldatei zuweisen

Mit diesem Befehl weisen Sie einem XSLT-Dokument eine **XML-Datei** zu. Der Befehl fügt eine Verarbeitungsanweisung ein, die eine XML-Datei spezifiziert, die mit dieser XSLT-Datei verarbeitet werden soll, wenn die XSL-Transformation an der XSLT-Datei ausgeführt wird:

```
<?altova_samplexml C:\workarea\html2xml\article.xml?>
```

Beachten Sie bitte, dass Sie den Pfad der zugewiesenen Datei relativ machen können. Aktivieren Sie dazu das Kontrollkästchen "Pfad relativ machen zu...".

15.7.9 Gehe zu XSL

Mit diesem Befehl öffnen Sie das zugehörige XSLT Dokument, falls Ihr XML-Dokument eine Processing Instruction zur Verarbeitung eines Stylesheet (also eine XSLT-Zuweisung) wie z.B. die folgende enthält:

© 2007 Altova GmbH

```
<?xml-stylesheet type="text/xsl" href="Company.xsl"?>
```

Der Befehl **Gehe zu XSL** öffnet das XSLT-Dokument anschließend in XMLSpy.

15.7.10 Debugger starten

Alt+F11

Mit diesem Befehl starten Sie den Debugger oder fahren mit der Verarbeitung des XSLT/XQuery-Dokuments fort. Wenn Breakpoints gesetzt wurden, wird der Ablauf an dieser Stelle angehalten. Wenn Tracepoints gesetzt wurden, wird das Ergebnis dieser Anweisungen im Fenster "Trace" angezeigt, wenn der schließende Node der Anweisung mit dem Tracepoint erreicht wurde. Wenn die Debugger-Sitzung noch nicht gestartet wurde, wird diese mit diesem Befehl gestartet und am ersten Node angehalten, um diesen zu verarbeiten. Wenn die Sitzung läuft, wird das XSLT/XQuery-Dokument bis zum Ende verarbeitet oder bis das Programm auf den nächsten Breakpoint stößt.

15.7.11 Debugger anhalten

Mit diesem Befehl stoppen Sie den Debugger. Dies ist nicht dasselbe, wie das Stoppen der Debugger-Sitzung, in der der Debugger gerade läuft. Dieser Befehl eignet sich, wenn Sie ein Dokument während einer Debugger-Sitzung bearbeiten oder andere Dateien in derselben Debugger-Sitzung verwenden möchten. Wenn Sie den Debugger angehalten haben, müssen Sie ihn erneut starten, um am Anfang des XSLT/XQuery-Dokuments zu beginnen.

15.7.12 Debugger neu starten

Mit diesem Befehl wird der XSLT Debugger mit den aktuell ausgewählten Dateien erneut gestartet und das Ausgabefenster wird gelöscht.

15.7.13 Debugger-Sitzung beenden

Mit diesem Befehl beenden Sie die Debugger-Sitzung und kehren zur normalen XMLSpy Ansicht zurück, die vor dem Start des Debuggers aktiv war. Ob die für die Debugger-Sitzung geöffneten Ausgabedokumente geöffnet bleiben oder nicht, hängt davon ab, welche Einstellung Sie im Dialogfeld XSLT/XQuery Debugger-Einstellungen gewählt haben.

15.7.14 Einsteigen

F11

Mit diesem Befehl gehen Sie Schritt für Schritt durch alle Nodes und XPath-Ausdrücke des Stylesheets. Dieser Befehl dient auch zum erneuten Starten des Debuggers, nachdem dieser angehalten wurde.

15.7.15 Aussteigen

Umschalt+F11

Mit diesem Befehl steigen Sie aus dem aktuellen Node aus und gehen zum nächsten **Zwilling** des Parent-Node oder zum nächsten Node auf der nächsthöheren Ebene über der des Parent-Node.

15.7.16 Überspringen

Strg+F11

Mit diesem Befehl überspringen Sie den aktuellen Node und gehen zum nächsten Node auf derselben Ebene oder zur nächsthöheren Ebene über der des aktuellen Node. Dieser Befehl dient auch zum erneuten Starten des Debuggers, nachdem er angehalten wurde.

15.7.17 Aktuellen Execution Node einblenden

Dieser Befehl dient zum Anzeigen/Auswählen des aktuellen Execution Node im XSLT/XQuery-Dokument und des entsprechenden Context Node im XML-Dokument. Dies ist nützlich, wenn Sie auf andere Register geklickt haben, auf denen bestimmter Code im XSLT Stylesheet bzw. in der XML-Datei angezeigt oder markiert wird, und anschließend wieder zum Ausgangspunkt zurückkehren möchten.

15.7.18 Breakpoint einfügen/löschen

F9

Dieser Befehl dient zum Einfügen oder Löschen eines Breakpoint an der angeklickten Position. Sie können sowohl für Nodes im XSLT/XQuery-Dokument als auch im XML-Dokument Breakpoints innerhalb einer Zeile definieren, um festzulegen, wo die Verarbeitung unterbrochen werden soll. An der Stelle, an der Sie den Breakpoint setzen, erscheint oberhalb des Nodes eine strichlierte rote Linie. An schließenden Nodes können keine Breakpoints definiert werden, und Breakpoints an Attributen in XSLT-Dokumenten werden ignoriert. Dieser Befehl steht auch zur Verfügung, wenn Sie mit der rechten Maustaste auf die gewünschte Stelle für den Breakpoint klicken.

15.7.19 Tracepoint einfügen/löschen

Umschalt+F9

Mit dem Befehl **XSL/XQuery | Tracepoint einfügen/löschen** wird ein Tracepoint an der aktuellen Cursorposition in einem XSLT/XQuery-Dokument eingefügt oder gelöscht. Bei Anweisungen mit einem Tracepoint wird der Wert der Anweisung beim Debuggen im Fenster "Trace" angezeigt, sobald der schließende Node dieser Anweisung erreicht wird. An der Stelle, an der Sie den Tracepoint setzen, erscheint oberhalb des Nodes eine strichlierte blaue Linie. An schließenden Nodes können keine Tracepoints definiert werden. Dieser Befehl steht auch zur Verfügung, wenn Sie mit der rechten Maustaste auf die gewünschte Stelle für den Breakpoint klicken.

© 2007 Altova GmbH

15.7.20 Breakpoint aktivieren/deaktivieren

Strg+F9

Mit diesem Befehl (es gibt keine Symbolleisten-Schaltfläche) schalten Sie bereits definierte Breakpoints ein oder aus. Die rote Breakpoint-Markierung wird grau angezeigt, sobald der Breakpoint deaktiviert wird. Der Debugger hält bei ausgeschalteten Breakpoints nicht an. Um einen Breakpoint ein- bzw. auszuschalten, setzen Sie den Cursor in den Node-Namen und klicken Sie auf **Breakpoint aktivieren/deaktivieren,** Dieser Befehl kann auch über das Kontextmenü aufgerufen werden, indem Sie mit der rechten Maustaste an die Stelle klicken, an der der Breakpoint aktiviert oder deaktiviert werden soll.

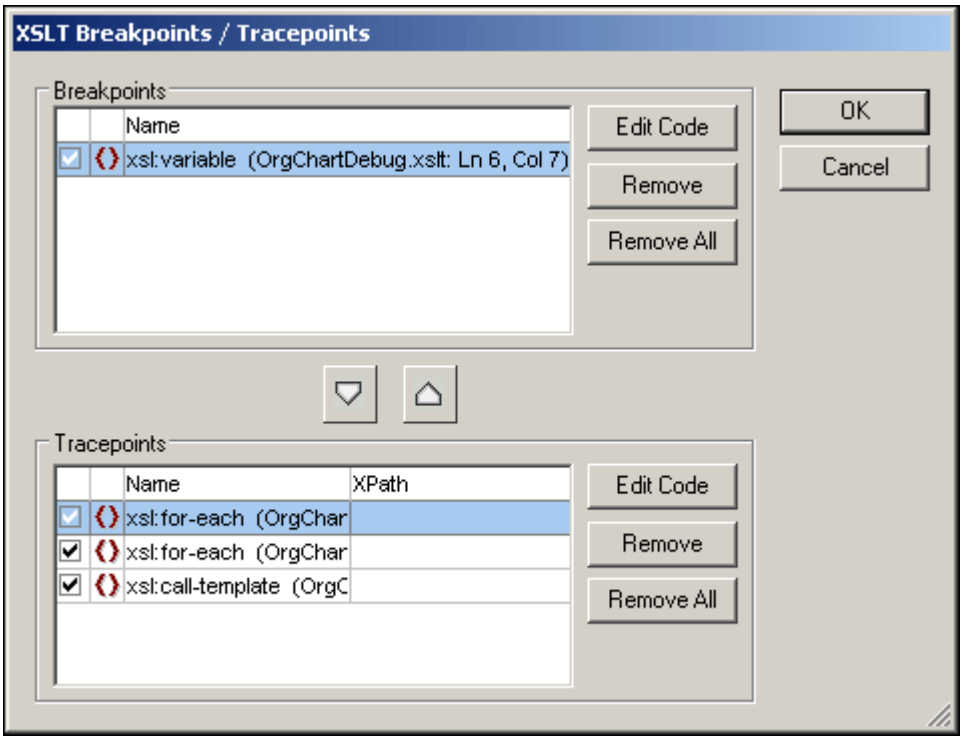
15.7.21 Tracepoint aktivieren/deaktivieren

Strg+Umschalt+F9

Mit diesem Befehl (es gibt keine Symbolleisten-Schaltfläche) schalten Sie bereits definierte Tracepoints ein oder aus. Die blaue Tracepoint-Markierung wird grau angezeigt, sobald der Tracepoint deaktiviert wird. Bei Anweisungen, bei denen der Tracepoint deaktiviert wurde, wird keine Ausgabe angezeigt. Um einen Tracepoint ein- bzw. auszuschalten, setzen Sie den Cursor in den Node-Namen und klicken Sie auf **Tracepoint aktivieren/deaktivieren,** Dieser Befehl kann auch über das Kontextmenü aufgerufen werden, indem Sie mit der rechten Maustaste an die Stelle klicken, an der der Tracepoint aktiviert oder deaktiviert werden soll.

15.7.22 Breakpoints/Tracepoints...

Mit diesem Befehl öffnen Sie das Dialogfeld "XSLT/XQuery Breakpoints", in welchem eine Liste zuvor definierter Breakpoints und Tracepoints (einschl. aller deaktivierten Breaktpoints und Tracepoints) in allen Dateien in der aktuellen Debugger-Sitzung angezeigt wird.



Das Kontrollkästchen zeigt an, ob ein Breaktpoint aktiv (Häkchen) oder deaktiviert ist. Um einen markierten Breakpoint/Tracepoint zu entfernen, klicken Sie auf die Schaltfläche **Entfernen.** Zum Entfernen aller Breakpoints klicken Sie auf **Alle löschen.** Wenn Sie auf die Schaltfläche **Code bearbeiten** klicken, wird der entsprechende Breakpoint in der Datei direkt aufgerufen.

Mit Hilfe des Abwärtspfeils können Sie den markierten Breakpoint in das Fenster **Tracepoints** verschieben. Mit dem Aufwärtspfeil können Sie den markierten Tracepoint in das Fenster **Breakpoints** verschieben.

In der Spalte **XPath** im Fenster "Tracepoints" können Sie einen XPath-Ausdruck für die einzelnen Tracepoints definieren.

15.7.23 Debug-Fenster

Wenn Sie den Cursor über den Befehl **Debug-Fenster** platzieren, erscheint ein Untermenü mit einer Liste der verschiedenen Informationsfenster des XSLT/XQuery-Debuggers. Wenn Sie eines der Informationsfenster aus dieser Liste auswählen, wird dieses Fenster auf der XSLT/XQuery Debugger-Oberfläche markiert. Dieser Befehl kann nur während einer laufenden Debugger-Sitzung verwendet werden.

15.7.24 XSLT/XQuery-Einstellungen

Mit diesem Befehl rufen Sie das Dialogfeld "XSLT/XQuery-Einstellungen" auf, in welchem Sie

 © 2007 Altova GmbH

Benutzereinstellungen für den Debugger festlegen können. Nähere Informationen dazu finden Sie im Abschnitt XSLT/XQuery Debugger.

15.8 Menü "Authentic"

Über die Authentic-Ansicht können Sie XML-Dokumente editieren, **die auf StyleVision Power Stylesheets basieren, die mit StyleVision von Altova erstellt worden sind!** Diese Stylesheets enthalten alle Informationen, die benötigt werden, damit eine XML-Datei grafisch in der Authentic-Ansicht dargestellt werden kann. Zusätzlich zu diesen Anzeigeinformationen ermöglichen Ihnen StyleVision Power Stylesheets außerdem, Daten in eine XML-Datei zu schreiben. Diese Daten werden mit Hilfe aller XSLT-Stylesheet-Funktionen dynamisch verarbeitet und sofort in Form einer Authentic-Ansicht ausgegeben.

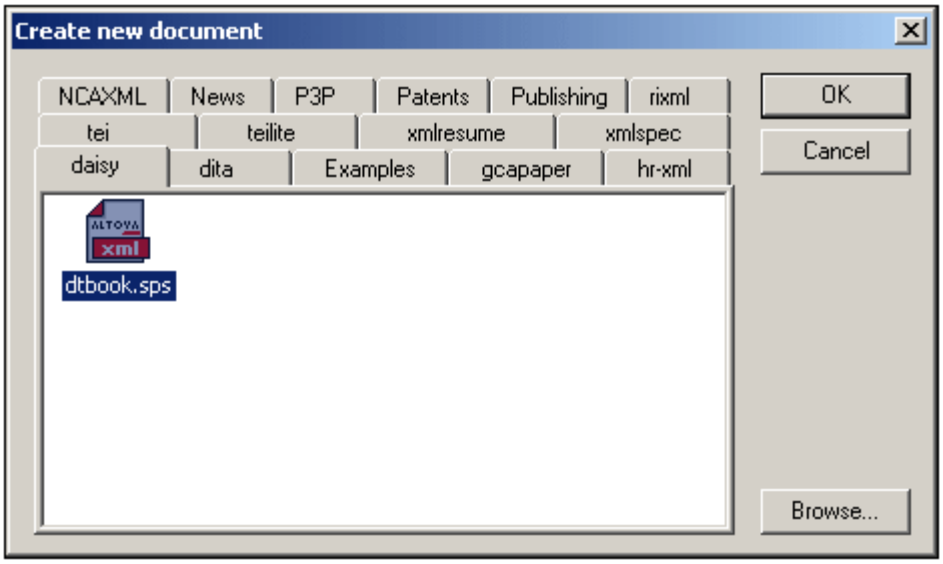
Des Weiteren können Sie StyleVision Power Stylesheets erstellen, um eine editierbare XML-Ansicht einer Datenbank anzuzeigen. Das StyleVision Power Stylesheet enthält Informationen, um eine Verbindung zu einer Datenbank herzustellen, die Datenbankdaten in der Authentic-Ansicht anzuzeigen und wieder zurück in die Datenbank zu schreiben.

Das Menü "Authentic" enthält Befehle zum Editieren von XML-Dokumenten in der Authentic -Ansicht. Ein Tutorial zur Authentic-Ansicht finden Sie im Abschnitt Tutorials.

15.8.1 Neues Dokument...

Mit dem Befehl **Neues Dokument...** können Sie eine neue XML-Dokumentenvorlage in der Authentic-Ansicht öffnen. Die XML-Dokumentenvorlage basiert auf einem StyleVision Power Stylesheet (`.sps`-Datei) und wird durch Auswahl des StyleVision Power Stylesheet geöffnet.

Wenn Sie den Befehl **Neues Dokument...** auswählen, wird das Dialogfeld "Neues Dokument anlegen" angezeigt.



Suchen Sie die gewünschte SPS-Datei und wählen Sie diese aus. Daraufhin wird eine XML-Dokumentenvorlage in der Authentic-Ansicht geöffnet.

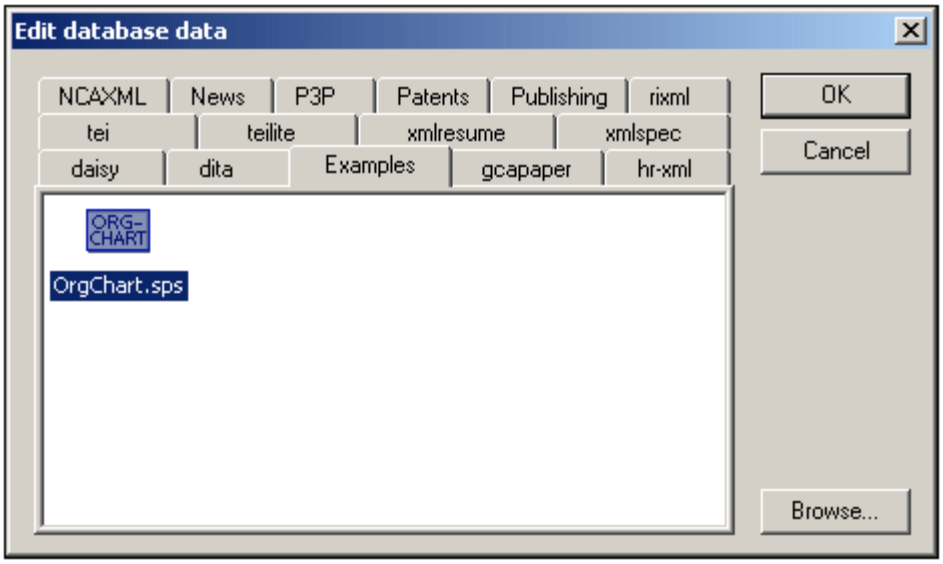
Bitte beachten Sie: StyleVision Power Stylesheets werden mit Hilfe von Altova <%SV-PROD%> erstellt. Dem StyleVision Power Stylesheet ist eine XML-Vorlagendatei zugewiesen. Die Daten in dieser XML-Datei liefern die Ausgangsdaten für die neue Dokumentenvorlage, die in der Authentic-Ansicht geöffnet wird.

© 2007 Altova GmbH

15.8.2 Datenbankdaten bearbeiten

Mit dem Befehl **Datenbankdaten bearbeiten...** öffnen Sie eine editierbare Ansicht einer Datenbank (DB) in der Authentic-Ansicht. Alle Informationen über die Herstellung einer Verbindung zur Datenbank, die Anzeige der DB und die Übernahme von Änderungen an der Datenbank über die Authentic-Ansicht befinden sich in einem StyleVision Power Stylesheet. Mit dem Befehl **Datenbankdaten bearbeiten...** öffnen Sie ein solches DB-basiertes StyleVision Power Stylesheet. Damit stellen Sie eine Verbindung zur DB her und zeigen die DB-Daten (durch eine XML-Linse) in der Authentic-Ansicht an.

Bei Aufruf des Befehls **Datenbankdaten bearbeiten...** erscheint das Dialogfeld "Datenbankdaten bearbeiten".



Suchen Sie die gewünschte SPS-Datei und wählen Sie diese aus. Diese Datei stellt die Verbindung zur DB her und öffnet eine editierbare Ansicht der Daten in der Authentic-Ansicht.

Warnung: Wenn Sie versuchen, ein StyleVision Power Stylesheet mit dem Befehl **Datenbankdaten bearbeiten...** zu öffnen, das nicht auf einer DB basiert, oder wenn Sie versuchen, ein DB-basiertes StyleVision Power Stylesheet zu öffnen, das in einer StyleVision -Version vor der Version StyleVision 2005 erstellt wurde, erhalten Sie eine Fehlermeldung.

Bitte beachten Sie: StyleVision Power Stylesheets werden mit Altova <%SV-PROD%> erstellt.

15.8.3 StyleVision Stylesheet zuweisen

Mit diesem Befehl weisen Sie einem **XML-Dokument** ein StyleVision Power Stylesheet (SPS) zu, um die Ansicht und Bearbeitung in der Authentic-Ansicht zu ermöglichen. Das StyleVision Power Stylesheet, das der XML-Datei zugewiesen werden soll, muss auf demselben Schema basieren, auf dem auch die XML-Datei basiert.

So weisen Sie einer XML-Datei ein StyleVision Power Stylesheet zu:

1. Machen Sie die XML-Datei zur aktiven und wählen Sie den Befehl **Stylevision Stylesheet zuweisen...**.
2. Daraufhin wird ein Dialogfeld geöffnet, in dem Sie angeben können, welche StyleVision Power Stylesheet-Datei zugewiesen werden soll.
3. Klicken Sie auf OK, um die gewünschte SPS-Anweisung in Ihr XML-Dokument

einzufügen. Beachten Sie bitte, dass Sie den Pfad der zugewiesenen Datei relativ machen können, indem Sie das Kontrollkästchen "Pfad relativ machen zu ..." aktivieren.

```
<?xml version="1.0" encoding="UTF-8"?>
<?altova_sps HTML-Orgchart.sps?>
```

Im obigen Beispiel ist der Name des StyleVision Power Stylesheet `HTML_Orgchart.sps`, und die Datei befindet sich im selben Verzeichnis wie die XML-Datei.

Bitte beachten Sie: In früheren Altova-Produktversionen wurde eine Verarbeitungsanweisung mit dem Ziel bzw. Namen `xmlspysps` verwendet. Eine Verarbeitungsanweisung sah daher etwa folgendermaßen aus: `<?xmlspysps HTML-Orgchart.sps?>`. Diese älteren Verarbeitungsanweisungen sind auch in aktuellen Altova-Produktversionen bei Authentic immer noch gültig.

15.8.4 StyleVision Stylesheet bearbeiten

Mit diesem Befehl wird StyleVision gestartet, so dass das StyleVision Power Stylesheet sofort editiert werden kann.

15.8.5 XML Entities definieren

Unabhängig davon, ob Ihr Dokument auf einer DTD oder einem XML-Schema basiert, können Sie für die Authentic-Ansicht Entitäten definieren, die dann in der Eingabehilfe "Entities" und im Kontext-Untermenü **Entity einfügen** angezeigt werden. Wenn Sie in der Eingabehilfe "Entities" auf eine Entität doppelklicken, wird die Entität an der aktuellen Cursorposition eingefügt.

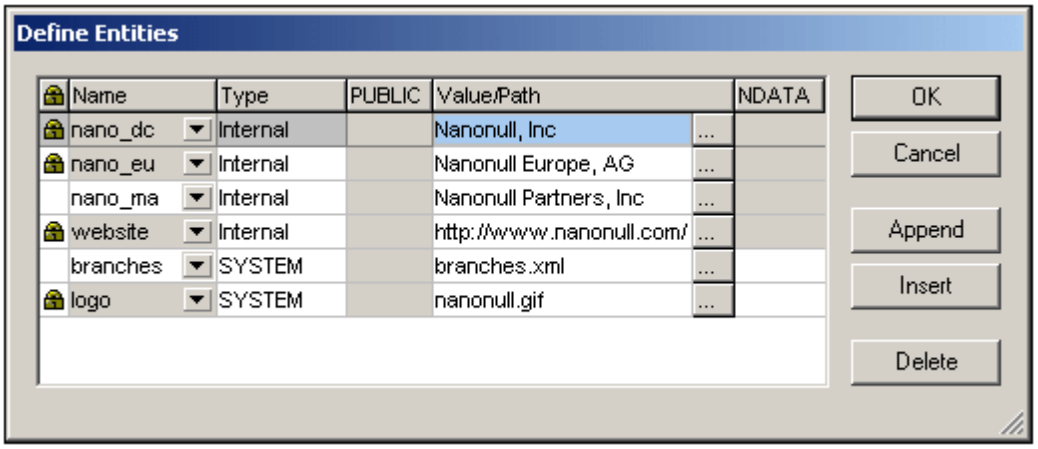
Eine Entität ist nützlich, wenn Sie einen Textstring, ein XML-Fragment oder eine andere externe Ressource an mehreren Stellen in Ihrem Dokument einfügen möchten. Sie definieren die Entität, die im Grund eine Kurzbezeichnung ist, die die gewünschten Daten repräsentiert, im Dialogfeld "Entities definieren". Nachdem Sie die Entität definiert haben, können Sie sie an mehreren Stellen in Ihrem Dokument einfügen. Dies spart Zeit und ist effizient, wenn Sie Änderungen am Dokument vornehmen möchten.

Sie können in Ihrem Dokument zwei Arten von Entitäten verwenden: **geparste** Entitäten, wobei es sich um XML-Daten handelt (entweder Textstrings oder ein Fragment eines XML-Dokuments) oder **nicht geparste** Entitäten, wobei es sich um nicht-XML-Daten wie z.B. eine Binärdatei handelt (gewöhnlich Grafiken, Sound oder Multimedia-Objekte). Jede Entität hat einen Namen und einen Wert. Bei geparsten Entitäten ist die Entität ein Platzhalter für die XML-Daten. Der Wert der Entität sind entweder die XML-Daten selbst oder eine URI, die auf eine XML-Datei verweist, die die XML-Daten enthält. Bei nicht geparsten Entitäten ist der Wert der Entität eine URI, die auf die nicht-XML-Datendatei verweist.

Um eine Entität zu definieren, gehen Sie vor wie folgt:

1. Wählen Sie **Authentic | XML Entities definieren...**. Dadurch wird das Dialogfeld "Entities definieren" geöffnet.

 © 2007 Altova GmbH



2. Geben Sie in das Feld "Name" den Namen der Entität ein. Dies ist der Name, der in der Eingabehilfe "Entities" angezeigt werden wird.
3. Wählen Sie im Feld "Type" den Entitätstyp aus der Dropdown-Liste aus. Es stehen drei Typen zur Auswahl. Bei einer **internen** Entität handelt es sich um Text, der im XML-Dokument selbst gespeichert ist. Bei Auswahl von **PUBLIC** oder **SYSTEM** geben Sie an, dass sich die Ressource außerhalb der XML-Datei befindet und mit Hilfe eines Public Identifier bzw. eines System Identifier aufgerufen wird. Ein System Identifier ist eine URI, die den Speicherort einer Ressource angibt. Der Public Identifier ist ein pfadunabhängiger Identifier, der es einigen Prozessoren erlaubt, die Ressource zu identifizieren. Wenn Sie sowohl einen Public Identifier als auch einen System Identifier definieren, wird der Public Identifier mit dem System Identifier aufgelöst und der System Identifier wird verwendet.
4. Wenn Sie als Typ PUBLIC ausgewählt haben, geben Sie den Public Identifier Ihrer Ressource im Feld PUBLIC ein. Wenn Sie Internal oder SYSTEM als Typ ausgewählt haben, wird das Feld PUBLIC deaktiviert.
5. Im Feld **Value/Path** haben Sie folgende Eingabemöglichkeiten:

   - Ist der Entitätstyp "Internal", geben Sie den gewünschten Textstring als Wert der Entität ein. Verwenden Sie zur Begrenzung der Entität keine Anführungszeichen, da diese als Teil des Textstrings behandelt werden.
   - Ist der Entitätstyp SYSTEM, geben Sie die URI der Ressource an oder wählen Sie mit Hilfe der Schaltfläche zum Durchsuchen eine Ressource in Ihrem lokalen Netzwerk aus. Wenn die Ressource geparste Daten enthält, muss es eine XML-Datei sein (d.h. sie muss eine XML-Erweiterung haben). Andernfalls kann als Ressource eine Binärdatei wie z.B. eine GIF-Datei verwendet werden.
   - Ist der Entitätstyp PUBLIC, müssen Sie in diesem Feld zusätzlich noch einen System Identifier eingeben.

6. Der Eintrag NDATA sagt dem Prozessor, dass diese Entität nicht geparst, sondern an den entsprechenden Prozessor gesendet werden soll. Das Feld NDATA sollte daher nur bei nicht geparsten Entitäten verwendet werden.

Dialogfeldfunktionen

Sie können Entitäten anhängen, einfügen und löschen, indem Sie auf die entsprechende Schaltfläche klicken. Des Weiteren können Sie Entitäten spaltenweise alphabetisch sortieren, indem Sie auf die Kopfzeile der jeweiligen Spalte klicken; bei einmaligem Klicken werden die Entitäten aufsteigend sortiert, bei zweimaligem Klicken absteigend. Die Größe des Dialogfelds und die Spaltenbreite lassen sich anpassen.

Sobald eine Entity im XML-Dokument verwendet wird, ist sie gesperrt und kann im Dialogfeld "Entities definieren" nicht mehr bearbeitet werden. Gesperrte Entities werden durch ein Schlosssymbol in der ersten Spalte gekennzeichnet. Durch das Sperren einer Entity wird

sichergestellt, dass das XML-Dokument hinsichtlich der Entities gültig ist. (Das Dokument wäre ungültig, wenn eine Entity referenziert ist, die nicht defininiert ist.)

Einschränkungen

- Eine in einer anderen Entität enthaltene Entität wird weder im Dialogfeld, noch der Authentic-Ansicht noch der XSLT-Ausgabe aufgelöst. Das kaufmännische Und-Zeichen einer solchen Entität wird in seiner Escape-Form also als `&amp angezeigt`.
- Externe Entitäten werden in der Authentic-Ansicht nicht aufgelöst, außer es handelt sich bei der Entität um eine Bilddatei, die als Wert eines Attributs eingegeben wurde, dessen Typ im Schema als `ENTITY` oder `ENTITIES` definiert wurde. Solche Entitäten werden aufgelöst, wenn das Dokument mit einem in der SPS generierten XSLT verarbeitet wird.

15.8.6 Markupsymbole ausblenden

Mit diesem Befehl werden die Markup-Symbole in der Authentic-Ansicht ausgeblendet.

15.8.7 Kleine Markup-Symbole einblenden

Mit diesem Befehl werden kleine Markup-Symbole in der Authentic-Ansicht angezeigt.

15.8.8 Große Markup-Symbole einblenden

Mit diesem Befehl werden große Markup Symbole in der Authentic-Ansicht angezeigt.

15.8.9 Gemischte Markup-Symbole einblenden

Mit diesem Befehl werden gemischte Markup Symbole in der Authentic-Ansicht angezeigt. Der Ersteller des StyleVision Power Stylesheet kann für die einzelnen Elemente/Attribute im Dokument entweder große, kleine oder gar keine Markup-Symbole definieren. Der Benutzer der Authentic-Ansicht sieht diese benutzerdefinierten Markup-Symbole im Ansichtsmodus für gemischte Markup-Symbole.

15.8.10 Zeile anhängen

Mit diesem Befehl wird eine Zeile an die Tabelle in der Authentic-Ansicht angehängt.

15.8.11 Zeile einfügen

© 2007 Altova GmbH

Mit diesem Befehl wird eine Zeile in die Tabelle in der Authentic-Ansicht eingefügt.

15.8.12 Zeile kopieren

Mit diesem Befehl wird die markierte Zeile in der Tabelle in der Authentic-Ansicht dupliziert.

15.8.13 Zeile nach oben

Mit diesem Befehl wird in der Authentic-Ansicht die markierte Zeile um eine Zeile nach oben verschoben.

15.8.14 Zeile nach unten

Mit diesem Befehl wird in der Authentic-Ansicht die markierte Zeile um eine Zeile nach unten verschoben.

15.8.15 Zeile löschen

Mit diesem Befehl wird in der Authentic-Ansicht die markierte Zeile aus der Tabelle gelöscht.

15.9 Menü "DB"

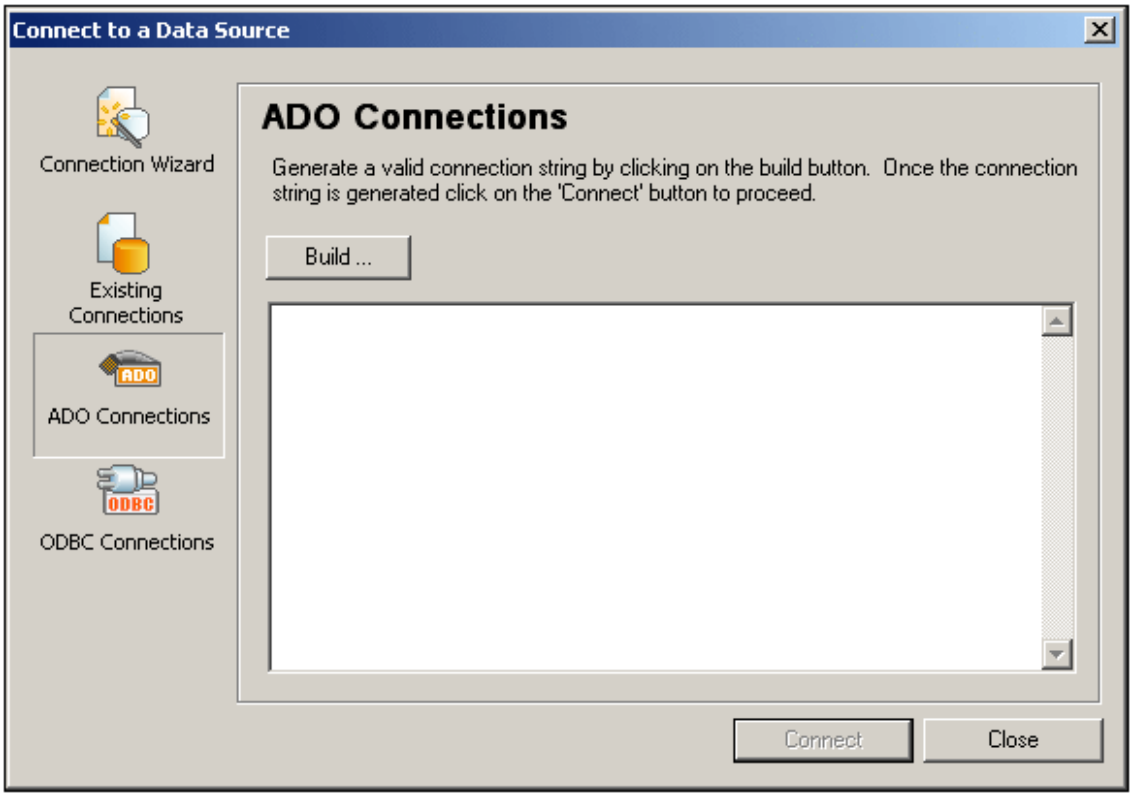
Das Menü **DB** enthält die folgenden Menüeinträge:

- Datenbank abfragen: Dient zum Abfragen verschiedener Datenbanken.
- IBM DB2: Enthält Befehle zur Unterstützung DB2-spezifischer Funktionalitäten.
- Tamino: Enthält Befehle für Tamino-Funktionalitäten
- Oracle-Datenbanken: Enthätl Befehle für die Arbeit mit Oracle-Datenbanken.

Da für alle in diesem Abschnitt beschriebenen Operationen eine Datenbankverbindung erforderlich ist, enthält dieser Abschnitt auch den Unterabschnitt Herstellen einer Verbindung zu einer Datenquelle.

15.9.1 Herstellen einer Verbindung zu einer Datenquelle

Für eine Reihe von Befehlen in den Menüs **DB** und **Konvertieren** muss eine Verbindung zu einer Datenbank hergestellt werden. XMLSpy unterstützt die Verbindung mit verschiedenen Datenbanken (*siehe Liste unten)*. Die Verbindung wird über das Dialogfeld "Verbindung zur Datenquelle herstellen" oder das Dialogfeld "Verbindungsassistent" (*Abbildung unten*) konfiguriert und hergestellt. In diesem Abschnitt finden Sie eine Beschreibung zur Funktionsweise dieses Dialogfelds. Bei Beschreibungen von Befehlen des Menüs **DB** oder **Konvertieren,** bei denen eine Verbindung zu einer Datenbank erforderlich ist, wird Bezug auf diese Dialogfelder genommen; in diesem Abschnitt finden Sie Informationen dazu, wie Sie mit Hilfe der Dialogfelder "Verbindung zur Datenquelle herstellen" oder "Verbindungsassistent" Datenbankverbindungen herstellen. Beide Dialogfelder sind fast identisch.



Die Optionen zum Herstellen einer Verbindung zur Datenbank sind:

- Über einen Verbindungsassistenten, der Sie durch den Vorgang führt.

- Über eine bestehende Verbindung
- Über ADO-Verbindungen
- Über ODBC-Verbindungen

Die einzelnen Optionen werden in den Unterabschnitten dieses Abschnitts näher beschrieben.

Unterstützte Datenbanken und ihre Root-Objekte
Es werden die folgenden Datenbanken unterstützt. Nachdem Sie eine Verbindung zur Datenbank hergestellt haben, stehen für die einzelnen unterstützten Datenbanken die folgenden Root-Objekte zur Verfügung:

Datenbank	Root-Objekt
MS SQL Server 2000, 2005	database
Oracle 9i und 10g	schema
MS Access 2000, 2003, 2007	database
MySQL 4.x und 5.x	database
IBM DB2 8.x und 9	schema
Sybase 11	database

Verbindungsassistent

Bei Auswahl der Schaltfläche "Verbindungsassistent" wird der Verbindungsassistent (*siehe Abbildung unten*) aufgerufen. Der Verbindungsassistent hilft Ihnen, eine Verbindung zu den gängigsten Datenbanktypen herzustellen. In diesem Abschnitt wird die Herstellung der Verbindung zu einer Microsoft Access Datenbank und einer IBM DB 2-Datenbank beschrieben.

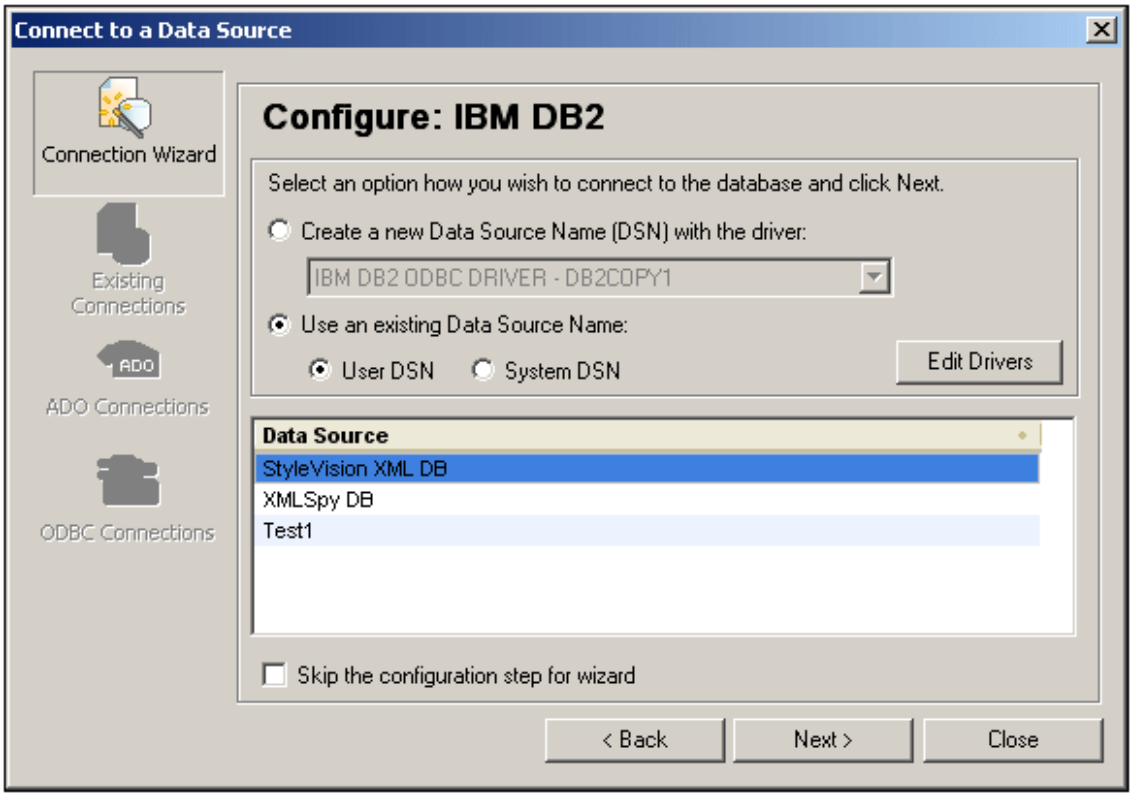
MS Access
Um eine Verbindung zu einer MS Access-Datenbank herzustellen, führen Sie die folgenden Schritte durch. Wählen Sie Microsoft Access (ADO) (*Abbildung unten*) und klicken Sie auf **Weiter**.

Suchen Sie im Dialogfeld "Verbindung zu MS Access herstellen" (*Abbildung unten*) die MS Access-Datenbank und klicken Sie auf **Weiter.** Die Verbindung zur Datenquelle wird hergestellt.

IBM DB2

Wählen Sie im ersten Bildschirm des Verbindungsassistenten (*siehe erste Abbildung in diesem Abschnitt*) den Eintrag IBM DB2 und klicken Sie auf **Weiter**. Daraufhin wird das Konfigurationsdialogfeld (*Abbildung unten*) angezeigt. Der Assistent führt Sie anschließend durch den Konfigurationsvorgang für die IBM DB2-Datenbank und stellt die Verbindung her. Dabei wird zuerst der entsprechende Treiber für die Verbindung mit der DB ausgewählt, anschließend die DB und schließlich werden die für die Verbindung erforderlichen Benutzerinformationen eingegeben.

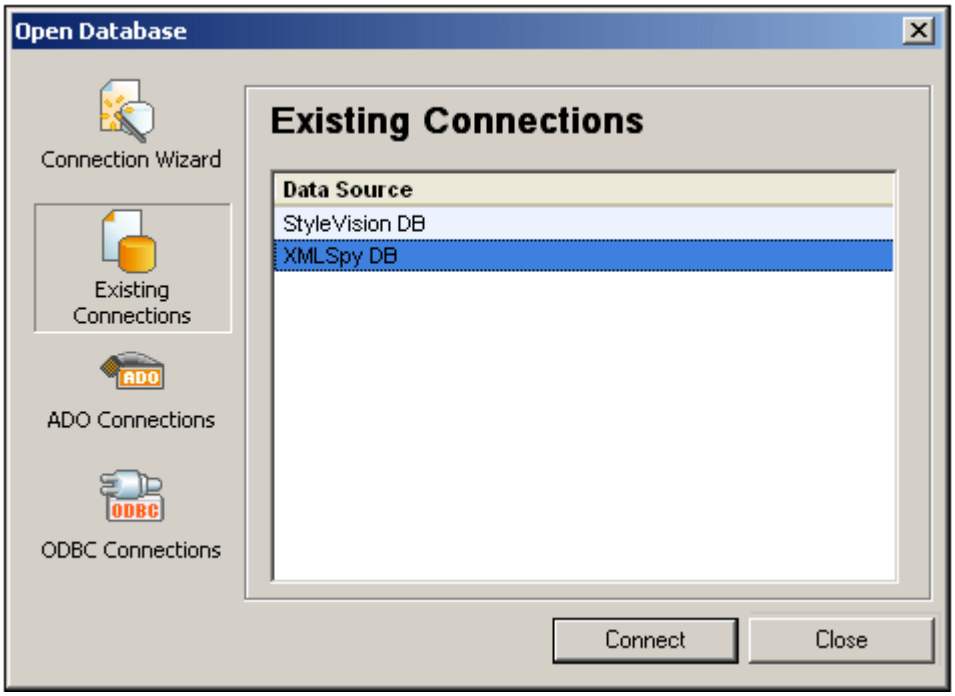
© 2007 Altova GmbH



Wählen Sie im Konfigurationsdialogfeld oben eine bestehende Datenquelle aus oder erstellen Sie eine neue Datenquelle mit einem entsprechenden Treiber (Zusätzliche Treiber können zur Liste der verfügbaren Treiber durch Klicken auf die Schaltfläche **Treiber bearbeiten** und Auswahl des gewünschten Treibers hinzugefügt werden). Im folgenden Bildschirm werden Sie aufgefordert, die Benutzerdaten (ID und Passwort) einzugeben. Wenn Sie auf **OK** klicken, wird die Verbindung zur Datenbank hergestellt.

Bestehende Verbindungen

Bei Auswahl der Schaltfläche "Bestehende Verbindungen" wird das Fenster "Bestehende Verbindungen" (*Abbildung unten*) geöffnet, in dem alle derzeit bestehenden Verbindungen angezeigt werden.



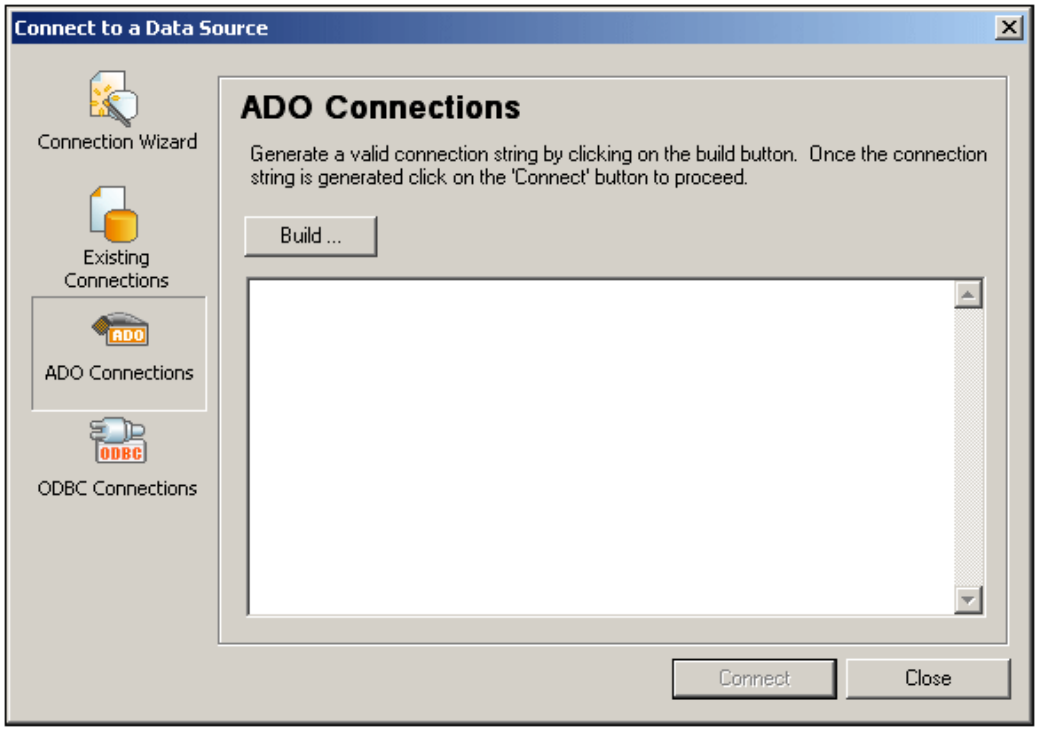
Wählen Sie die gewünschte Datenbank aus und klicken Sie auf **Verbindung herstellen**. Die Verbindung zur Datenbank wird hergestellt.

ADO-Verbindungen

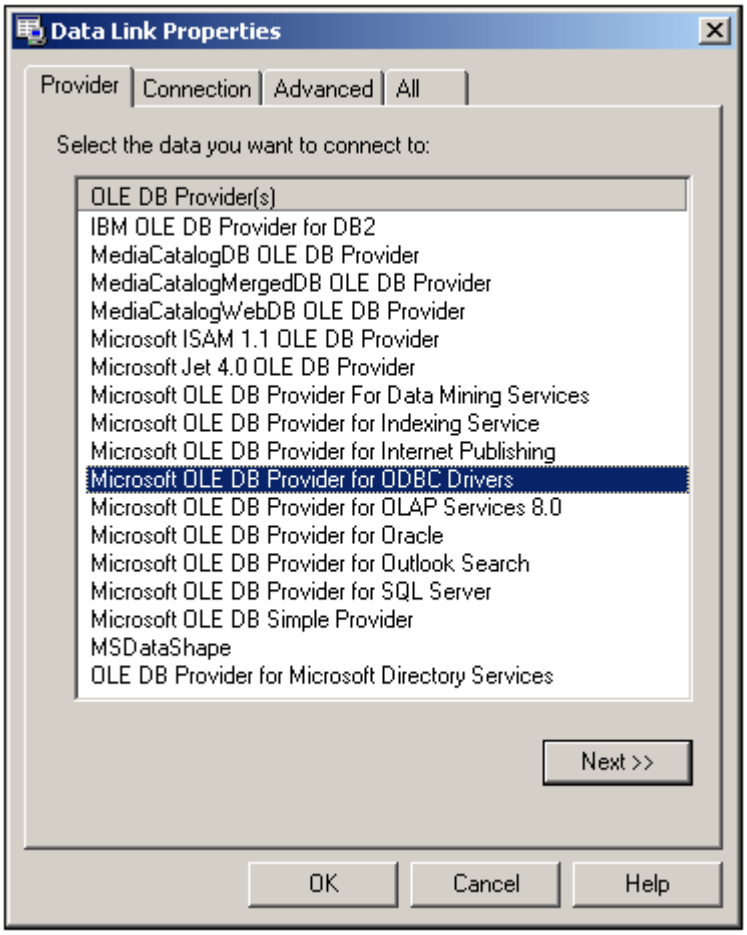
Hier wird anhand eines Beispiels mit IBM DB2 erklärt, wie eine ADO-Verbindung hergestellt wird.

1. Wählen Sie im Dialogfeld "Verbindung zur Datenquelle herstellen" die Option "ADO-Verbindungen" aus. Daraufhin wird das Dialogfeld "ADO-Verbindungen" angezeigt (*Abbildung unten*).

© 2007 Altova GmbH

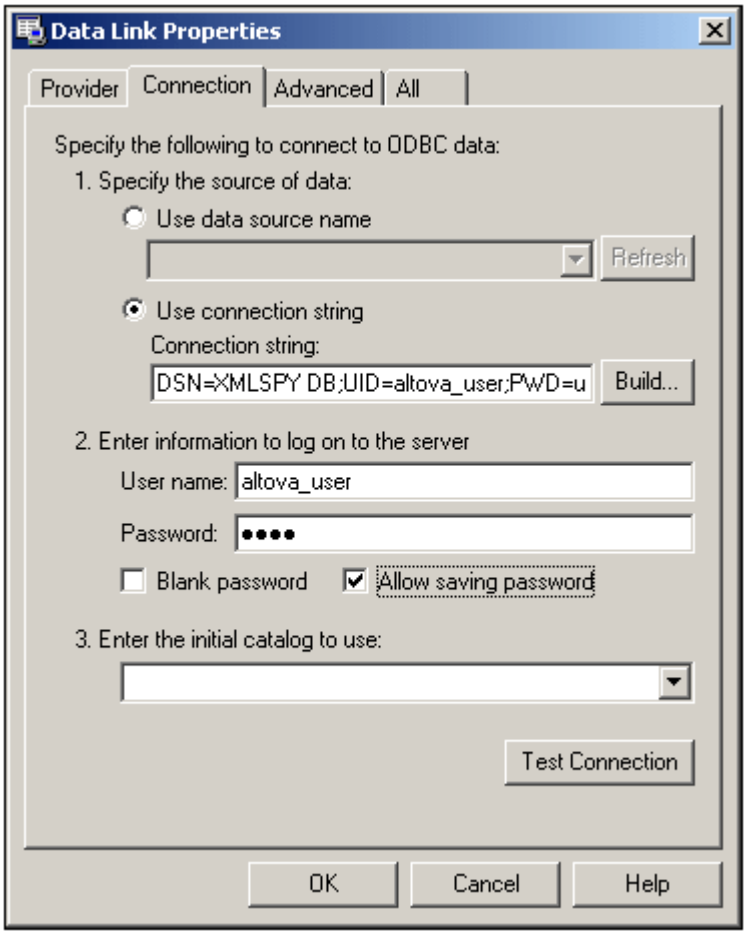


2. Klicken Sie auf die Schaltfläche **Erzeugen**. Das Dialogfeld "Datenverknüpfungseigenschaften" (*Abbildung unten*) wird geöffnet.

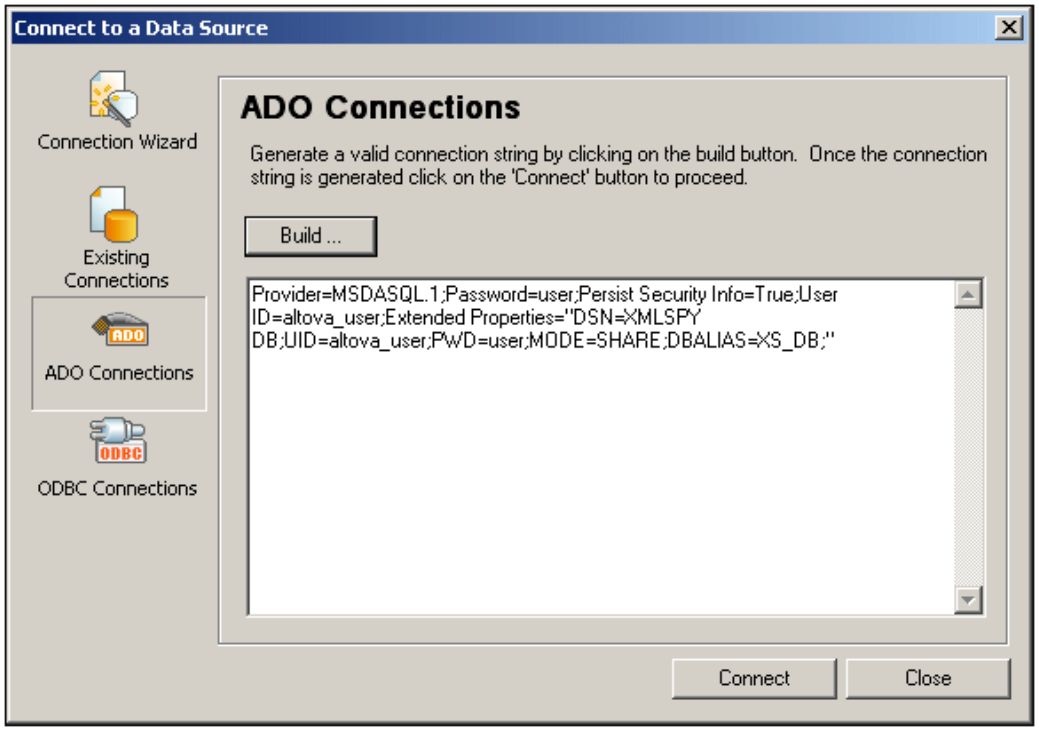


3. Wählen Sie "Microsoft OLE DB Provider for ODBC Drivers" in der Drowdown-Liste aus und klicken Sie auf **Weiter**. Das Register "Verbindung" ist aktiv.

© 2007 Altova GmbH



4. Geben Sie über die Schaltfläche **Erzeugen** einen Connection String ein. (In dem Dialogfeld, das erscheint, werden Sie aufgefordert die Datenquelle und Benutzerdaten einzugeben).
5. Falls Login-Informationen benötigt werden, geben Sie den Benutzernamen und das Passwort ein und aktivieren Sie das Kontrollkästchen "Speichern des Kennworts zulassen".
6. Klicken Sie auf **OK**. Das Dialogfeld "Datenverknüpfungseigenschaftten" wird geschlossen. Der Connection String wird im Feld "ADO-Verbindungen" angezeigt (*Abbildung unten*).



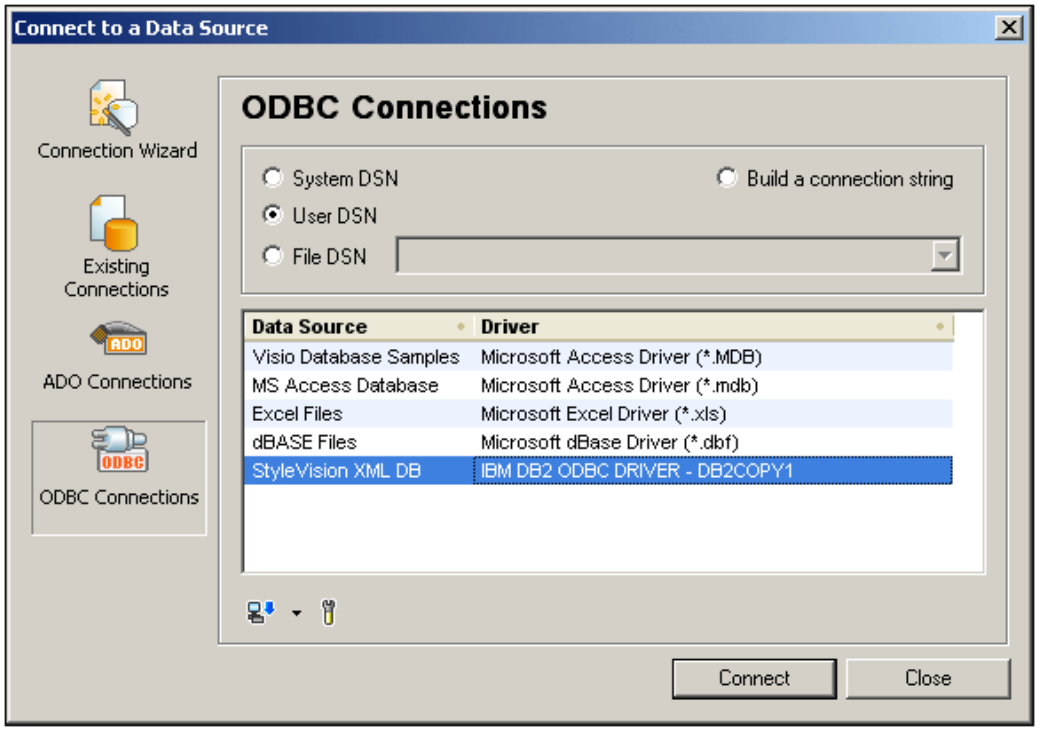
7. Klicken Sie auf die Schaltfläche **Verbindung herstellen**. Daraufhin wird eine Verbindung zur Datenquelle hergestellt.

ODBC-Verbindungen

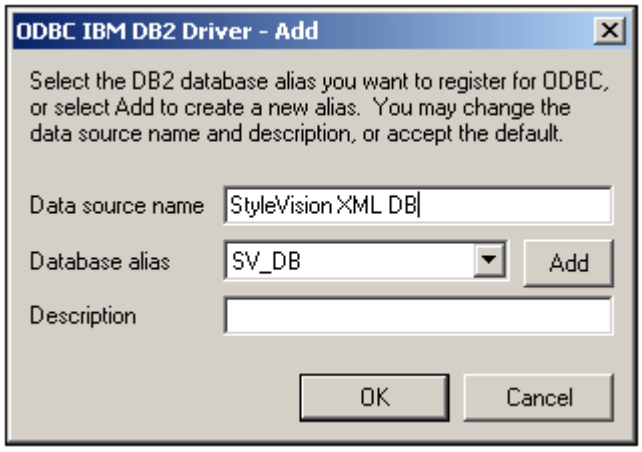
Um eine Verbindung zu einer IBM DB2-Datenbank über ODBC herzustellen, gehen Sie folgendermaßen vor:

1. Wählen Sie die Option **ODBC-Verbindungen** aus.

© 2007 Altova GmbH



2. Wählen Sie eine der Optionen zur Definition eines Datenquellennamens (DSN). Wenn Sie "System-DSN" oder "Benutzer-DSN" wählen, werden die verfügbaren DSNs im Fenster "Datenquelle" angezeigt. Bei Auswahl von "Datei-DSN" werden Sie aufgefordert, den Ordner auszuwählen, der die Datei-DSNs enthält. Alternativ dazu können Sie einen Connection String zu DB erzeugen, indem Sie die Option "Connections String erstellen" auswählen.
3. Wenn Sie einen DSN zu den DSNs in der Liste der Datenquellen hinzufügen möchten, gehen Sie folgendermaßen vor: (i) Klicken Sie auf den Dropdown-Pfeil der Schaltfläche "Neue Datenquelle erstellen"; (ii) wählen Sie den gewünschten Treiber aus; (iii) wählen Sie im Dialogfeld, das daraufhin angezeigt wird (*Abbildung unten*), den DB-Alias aus und geben Sie ihm einen DSN.

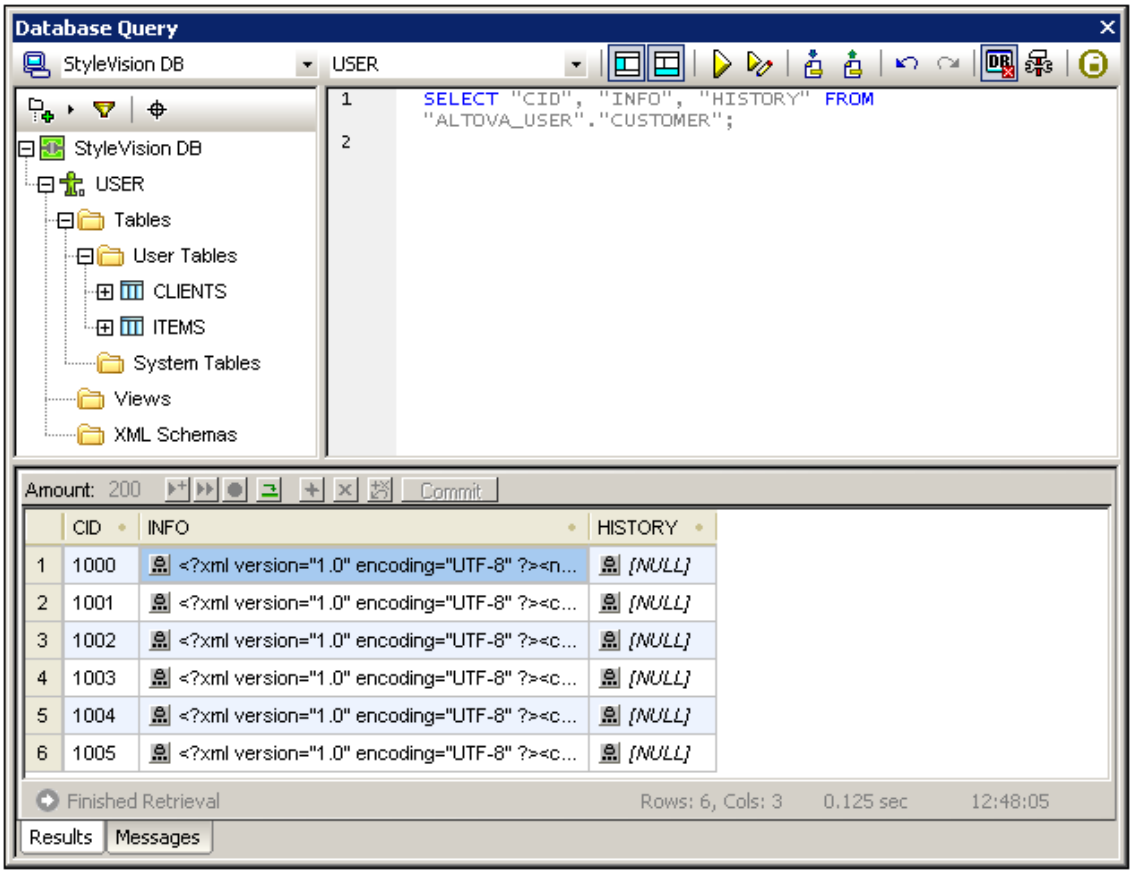


(iv) Klicken Sie zum Fertigstellen auf **OK.** Die DB wird als Datenquelle zur Liste im Bereich "Datenquelle" hinzugefügt.
4. Wählen Sie im Bereich "Datenquelle" die Datenquelle aus und klicken Sie auf **Verbindung herstellen**.

5. Wenn Sie nach Ihrer Benutzer-ID und dem Passwort gefragt werden, geben Sie diese ein und klicken Sie auf **OK**. Die Verbindung zur Datenquelle wird hergestellt.

15.9.2 Datenbank abfragen

Mit dem Befehl **Datenbank abfragen** wird das Fenster "Datenbankabfrage" (*Abbildung unten*) geöffnet. Sobald dieses Fenster geöffnet wurde, können Sie es durch Auswahl des Befehls **DB | Datenbank abfragen** bzw. durch Klicken auf die Symbolleisten-Schaltfläche "Datenbank abfragen" ein- und ausblenden.



Übersicht über das Fenster "Datenbankabfrage"
Das Fenster "Datenbankabfrage" besteht aus drei Teilen:

- einem **Browser-Fenster** links oben, in dem Verbindungsinformationen und Datenbanktabellen angezeigt werden.
- einem **Abfragefenster** rechts oben, in das die Abfrage eingegeben wird.
- einem mit Registern versehenen **Ergebnis-/Meldungsfenster**. Im Fenster "Ergebnisse" werden die Ergebnisabfrage im sogenannten Ergebnisraster angezeigt. Im Fenster "Meldungen" werden die Meldungen (einschl. Fehler- und Warnmeldungen), die bei der Ausführung der DB-Abfrage generiert werden, angezeigt.

Das Fenster "Datenbankabfrage" verfügt im oberen Bereich über eine Symbolleiste. Zwei der Schaltflächen in dieser Symbolleiste sind im Folgenden beschrieben. Die restlichen Schaltflächen werden im Abschnitt Fenster "Abfrage": Beschreibung und Funktionen beschrieben.

Blendet das Browser-Fenster ein bzw. aus.

Blendet das Ergebnis-/Meldungsfenster ein bzw. aus.

Übersicht über die Mechanismen zur Abfrage von Datenbanken
Die Vorgangsweise bei der Abfrage von Datenbanken ist die folgende. Sie wird in den Unterabschnitten dieses Abschnitts näher beschrieben.

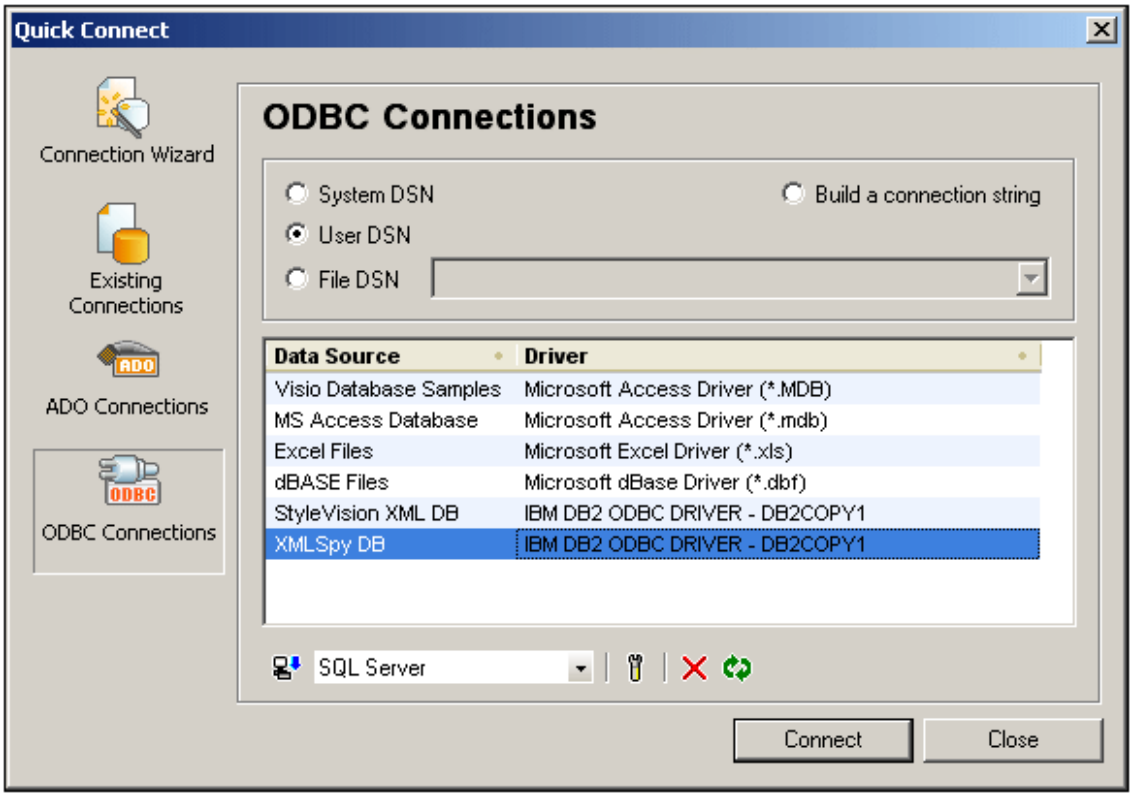
1. Über das Fenster "Datenbankabfrage" wird eine Verbindung zur Datenbank hergestellt. Unterstützt werden die Datenbanken: MS Access 2000 und 2003; Microsoft SQL Server; Oracle; MySQL; Sybase; und IBM DB2.
2. Die Datenbank, zu der eine Verbindung hergestellt wurde, bzw. Teile dieser Datenbank werden im Browser-Fenster angezeigt. Dieses Fenster kann je nach gewünschter Ansicht angepasst werden.
3. Im Abfragefenster wird eine Abfrage gemäß den für die gewünschte Datenbank benötigten Syntaxregeln eingegeben und ausgeführt.
4. Die Ergebnisse der Abfrage können mit verschiedenen Filtern gefiltert werden, bearbeitet und auf andere Arten verwendet werden.

Datenquellen

Zur Ausführung einer Datenbankabfrage müssen Sie zuerst eine Verbindung zur gewünschten Datenbank herstellen. In diesem Abschnitt wird beschrieben, wie Sie:

- eine Verbindung zu einer Datenbank herstellen und
- die gewünschte Datenquelle und das Root-Objekt aus mehreren bestehenden Verbindungen auswählen.

Herstellen einer Verbindung zu einer Datenbank
Wenn Sie zum ersten Mal in einer Sitzung (oder wenn keine Datenbankverbindung besteht) auf den Befehl **Datenbank abfragen** klicken, wird das Dialogfeld "Verbindungsassistent" (*Abbildung unten*) aufgerufen, über das Sie eine Verbindung zu einer Datenbank herstellen können. Um später weitere Verbindungen herzustellen, klicken Sie im Fenster "Datenbankabfrage" auf die Schaltfläche "Verbindungsassitent" .



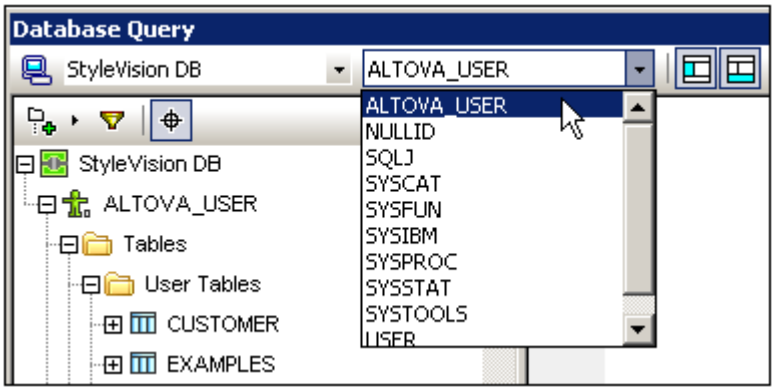
Eine Beschreibung zum Herstellen einer Verbindung zu einer Datenbank über das Dialogfeld "Verbindungsassistent" wird im Abschnitt Herstellen einer Verbindung zu einer Datenquelle beschrieben.

Unterstützte Datenbanken und ihre Root-Objekte
Es werden die folgenden Datenbanken unterstützt. Nachdem Sie eine Verbindung zur Datenbank hergestellt haben, stehen für die einzelnen unterstützten Datenbanken die folgenden Root-Objekte zur Verfügung:

Datenbank	Root-Objekt
MS SQL Server 2000, 2005	database
Oracle 9i und 10g	schema
MS Access 2000, 2003, 2007	database
MySQL 4.x und 5.x	database
IBM DB2 8.x und 9	schema
Sybase 11	database

Auswahl der erforderlichen Datenquelle
Alle bestehenden Verbindungen bzw. Root-Objekte werden in zwei Auswahllisten in der Symbolleiste des Fensters "Datenbankabfrage" (*Abbildung unten*) aufgelistet.

© 2007 Altova GmbH



In der Abbildung oben wurde die Datenbank mit dem Namen `StyleVision DB` ausgewählt. Von den verfügbaren Root-Objekten für diese Datenbank wurde das Root-Objekt `ALTOVA_USER` ausgewählt. Die Datenbank und das Root-Objekt werden anschließend im Browser-Fenster angezeigt.

Browser-Fenster: Anzeigen der DB-Objekte

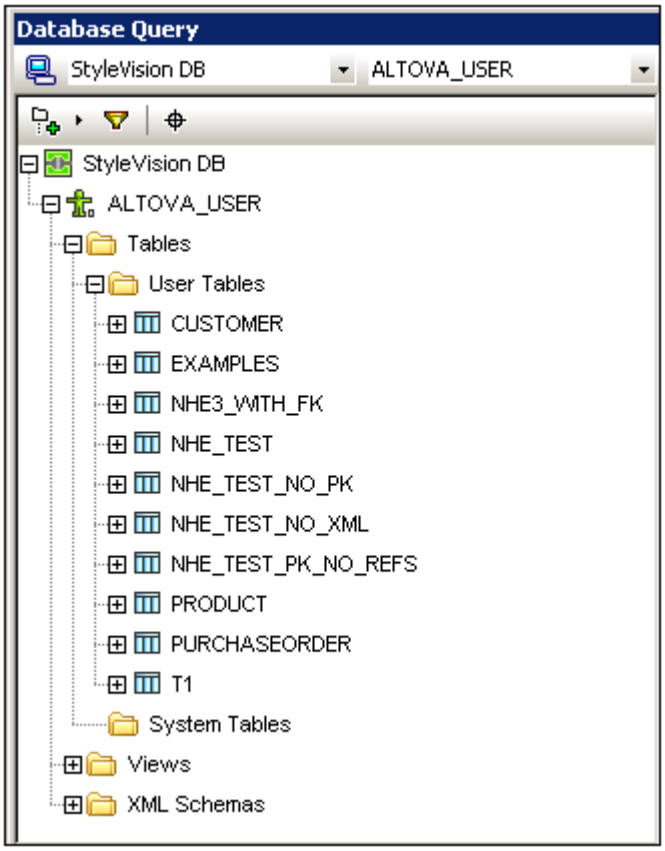
Im Browser-Fenster sehen Sie eine Übersicht über die Objekte in der ausgewählten Datenbank. Diese enthält Datenbank-Constraint-Informationen, z.B. ob es sich bei einer Spalte um einen Primär- oder Sekundärschlüssel handelt. In Datenbanken vom Typ IBM DB2 Version 9 sehen Sie im Browser zusätzlich registrierte XML-Schemas in einem separaten Ordner.

In diesem Abschnitt wird Folgendes beschrieben:

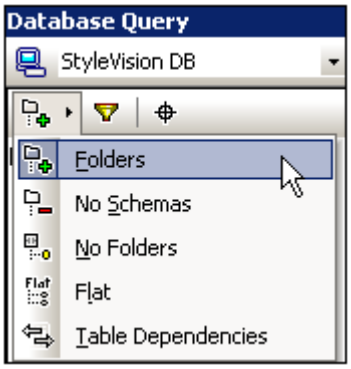
- Die im Browser-Fenster verfügbaren Layouts.
- Wie Datenbankobjekte gefiltert werden.
- Wie Datenbankobjekte gesucht werden.

Browser-Fenster Layouts

Im Standard-Ordner-Layout werden Datenbankobjekte hierarchisch angezeigt. Je nachdem, welches Objekt ausgewählt ist, stehen unterschiedliche Kontextmenüoptionen zur Verfügung, wenn Sie mit der rechten Maustaste auf ein Objekt klicken.



Um ein Layout für den Browser auszuwählen, klicken Sie in der Symbolleiste des Browser-Fensters auf die Schaltfläche "Layout" und wählen Sie das Layout aus der Dropdown-Liste aus (*Abbildung unten*). Beachten Sie, dass sich die Schaltfläche ändert, je nachdem, welches Layout Sie ausgewählt haben.



Die verfügbaren Layouts sind:

- *Ordner:* Gliedert die Datenbankobjekte auf Basis des Objekttyps in hierarchisch angelegten Ordnern. Dies ist die Standardeinstellung.
- *Keine Schemas:* Ähnlich dem Ordner-Layout, mit der Ausnahme, dass es keine Ordner für Datenbank-Schemas gibt; Tabellen werden daher nicht nach dem Datenbankschema gegliedert.
- *Keine Ordner:* Zeigt die Datenbankobjekte in einer Hierchie an, ohne Ordner zu verwenden.
- *Flach:* Gliedert die Datenbankobjekte auf der ersten hierarchischen Ebene nach Typ.

© 2007 Altova GmbH

Anstatt dass Spalten z.B. in der entsprechenden Tabelle angezeigt werden, werden alle Spalten in einem separaten Spaltenordner angezeigt.

- *Tabellenabhängigkeiten:* Kategorisiert die Tabellen nach ihrer Beziehung zu anderen Tabellen. Es gibt Kategorien für Tabellen mit Sekundärschlüsseln, Tabellen, die durch Sekundärschlüssel referenziert werden und Tabellen, die keine Beziehung zu anderen Tabellen haben.

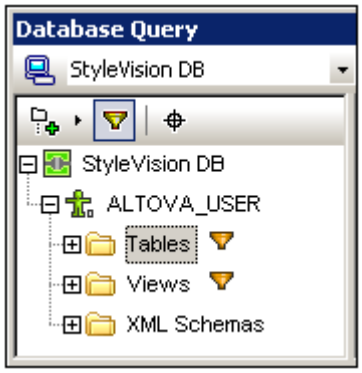
Um Tabellen nach Benutzer- und Systemtabellen zu sortieren, wechseln Sie ins Layout "Ordner", "Keine Schemas" oder "Flach", rechtsklicken Sie auf die Tabelle und wählen Sie den Befehl **Nach Benutzer- und Systemtabellen sortieren**. Die Tabellen werden in den Ordner "Benutzertabellen" und "Systemtabellen" in alphabetischer Reihenfolge sortiert.

Filtern von Datenbankobjekten

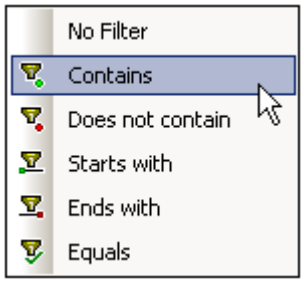
Im Browser-Fenster können Schemas, Tabellen und Ansichten (in allen Layouts mit Ausnahme des Layouts "Keine Ordner") nach Namen oder dem Teil eines Namens gefiltert werden. Die Objekte werden während der Eingabe der Zeichen gefiltert, dabei wird die Groß- und Kleinschreibung standardmäßig nicht beachtet.

So filtern Sie Objekte im Browser:

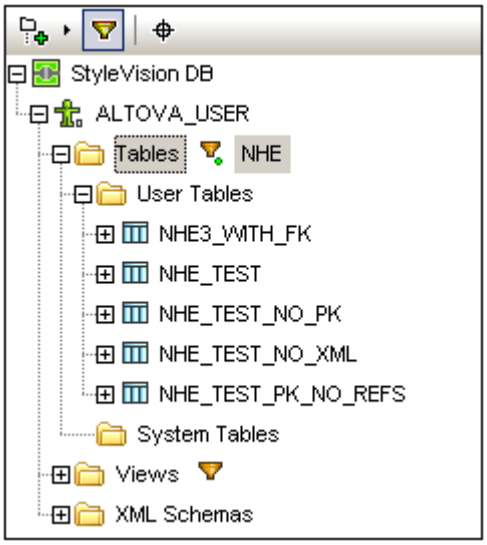
1. Klicken Sie in der Symbolleiste des Browser-Fensters auf die Schaltfläche "Ordnerinhalt filtern". Neben allen Ordnern im ausgewählten Layout erscheinen Filtersymbole (*Abbildung unten*).



2. Klicken Sie auf das Filtersymbol neben dem gewünschten Ordner und wählen Sie aus dem Popup-Menü die Filteroption aus, z.B. `Enthält`.

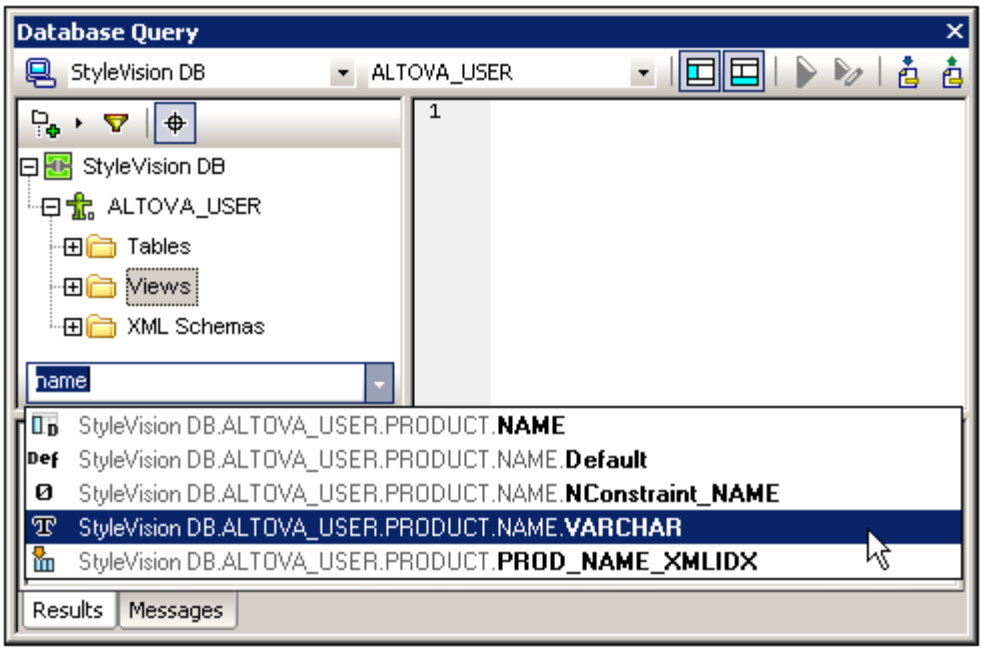


3. Geben Sie im Eingabefeld, das nun angezeigt wird, den Filterstring ein (in der Abbildung unten ist der Filterstring für den Ordner `Tables` der String `NHE`). Der Filter wird während der Eingabe angewendet.



Suchen von Datenbankobjekten
Um ein bestimmtes Datenbankobjekt nach seinem Namen zu suchen, können Sie die Objektsuche des Browser-Fensters verwenden. Dies funktioniert folgendermaßen:

1. Klicken Sie in der Symbolleiste des Browser-Fensters auf die Schaltfläche "Objektsuche". Eine Dropdown-Liste erscheint am unteren Rand des Browsers.
2. Geben Sie den Suchstring, z.B. `name` in das Eingabefeld dieser Liste ein (*Abbildung unten*). Wenn Sie auf den Dropdown-Pfeil klicken, werden alle Objekte, die den Suchstring enthalten, angezeigt.



3. Klicken Sie auf das Objekt in der Liste, um es im Browser zu sehen.

 © 2007 Altova GmbH

Abfragefenster: Beschreibung und Funktionen

Das Abfragefenster ist ein intelligenter SQL Editor zur Eingabe von Abfragen für die ausgewählte Datenbank. Klicken Sie nach Eingabe der Abfrage im Fenster "Datenbankabfrage" auf die Schaltfläche "Ausführen", um die Abfrage auszuführen. Die Ergebnisse und die Ausführungsmeldungen werden im Ergebnis-/Meldungsfenster angezeigt. Eine Anleitung zum Arbeiten mit Abfragen finden Sie im nächsten Abschnitt, Abfragefenster: Arbeiten mit Abfragen. In diesem Abschnitt werden die Hauptfunktionen des Abfragefensters beschrieben.

- SQL Editor-Schaltflächen in der Datenbankabfrage-Symbolleiste
- SQL Editor-Optionen
- Autokomplettierung von SQL-Anweisungen
- Definition von Regionen in einem SQL Script
- Einfügen von Kommentaren in ein SQL-Script
- Verwendung von Lesezeichen

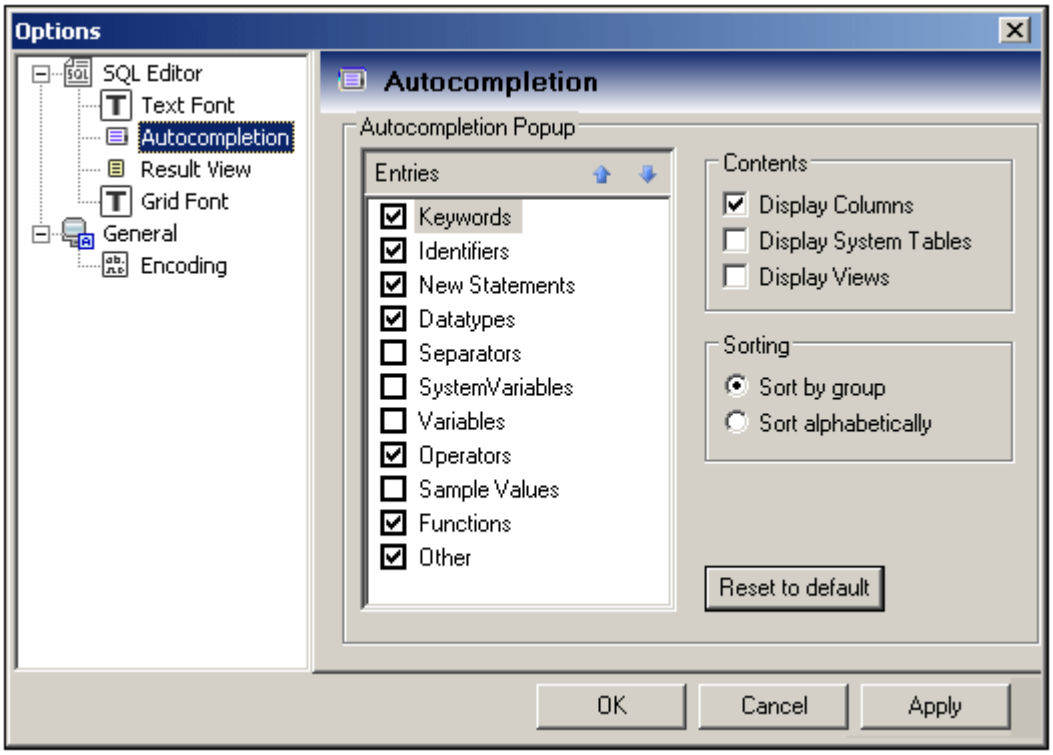
SQL Editor-Schaltflächen in der Datenbankabfrage-Symbolleiste

Für die Arbeit mit dem SQL Editor werden die folgenden Schaltflächen in der Symbolleiste des Fensters "Datenbankabfrage" verwendet:

	Ausführen	Führt die aktuell ausgewählte SQL-Anweisung aus. Wenn das Script mehrere Anweisungen enthält und keine ausgewählt wird, werden alle ausgeführt.
	Zur Datenbankbearbeitung ausführen	Wie der Befehl "Ausführen", mit der Ausnahme, dass die Ergebnisse (auf dem Register "Ergebnisse") bearbeitet werden können.
	SQL-Datei importieren	Öffnet eine SQL-Datei im SQL Editor.
	SQL-Datei exportieren	Speichert SQL-Abfragen in einer SQL-Datei.
	Rückgängig	Macht unbegrenzt viele Bearbeitungen im SQL Editor rückgängig.
	Wiederherstellen	Stellt unbegrenzt viele Bearbeitungen im SQL Editor wieder her.
	Optionen	Öffnet das Dialogfeld "Optionen" im SQL Editor.
	SQL-Script in DatabaseSpy öffnen	Öffnet das SQL Script in Altova DatabaseSpy.

SQL Editor Optionen

Wenn Sie in der Datenbankabfrage-Symbolleiste auf die Schaltfläche "Optionen" klicken, wird das Dialogfeld "Optionen" angezeigt (*Abbildung unten*).



Die wichtigsten Einstellungen sind die folgenden:

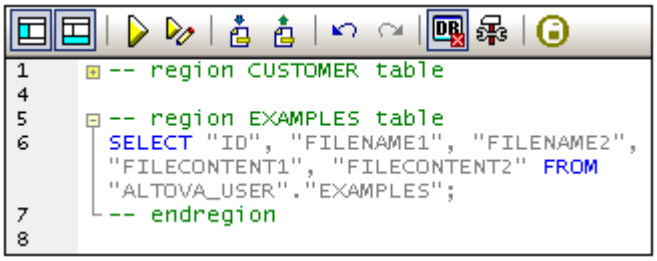
- ***SQL Editor | Textschriftart:*** Optionen zum Einstellen der Schriftart des Texts im SQL Editor.
- ***SQL Editor | Autokomplettierung:*** Wenn die Autokomplettierungsfunktion (auf dem Register "SQL Editor") aktiviert ist, können Sie auswählen, welche Art von Einträgen die Autokomplettierung unterstützen soll (Register "Autokomplettierung", *Abbildung oben*). Die Autokomplettierungsfunktion bietet während der Eingabe eine Liste relevanter Einträge zur Auswahl. Sie steht für die folgenden Datenbanken zur Verfügung: MS SQL Server 2000 und 2005, MS Access 2003 und IBM DB2 v.9. In der Gruppe *Inhalt* können Sie definieren, ob Spalten, Systemtabellen oder Ansichten im Autokomplettierungs-Popup-Fenster angezeigt werden sollen. In der Gruppe *Sortierung* können Sie auswählen, ob Einträge nach Gruppe oder alphabetisch sortiert angezeigt werden sollen.
- ***SQL Editor | Ergebnisansicht:*** Optionen zum Konfigurieren des Ergebnisregisters.
- ***SQL Editor | Rasterschriftart:*** Optionen zum Einstellen der Schriftart für den Text auf dem Register "Ergebnisse" im Ergebnisraster.
- ***Allgemein | Kodierung:*** Optionen zum Einstellen der Kodierung für neue SQL-Dateien und bestehende SQL-Dateien, bei denen die Kodierung nicht erkannt wird. (Wenn die Kodierung bestehender SQL-Dateien erkannt wird, werden die Dateien geöffnet und gespeichert, ohne dass die Kodierung geändert wird.)

Autokomplettierung von SQL-Anweisungen
Um die Autokomplettierungsfunktion zu aktivieren, aktivieren Sie im Dialogfeld "Optionen" auf dem Register "SQL Editor" das Kontrollkästchen "Autokomplettierung". Wenn diese Funktion aktiv ist, erscheinen während der Eingabe von SQL-Anweisungen Popup-Fenster (*Abbildung unten*).

 © 2007 Altova GmbH

Drücken Sie die **Eingabetaste** um den markierten Eintrag einzufügen.

Definition von Regionen in einem SQL Script
Regionen sind Abschnitte in SQL Scripts, die als Einheit markiert und deklariert sind. Regionen können erweitert und reduziert werden, um Teile von SQL Scripts ein- oder auszublenden. Regionen können auch in andere Regionen verschachtelt werden. Regionen werden werden vor und nach dem ausgewählten Text durch den Kommentar `--region` bzw. `--endregion` gekennzeichnet. Regionen können auch optional einen Namen erhalten, der nach der `--region` Markierung eingegeben wird (*siehe Abbildung unten*).



Um eine Region einzufügen, wählen Sie die gewünschte(n) Anweisungen aus, die zu einer Region gemacht werden sollen, rechstklicken Sie und wählen Sie den Befehl "Region einfügen". Es wird eine erweiterbare/reduzierbare Region erstellt. Fügen Sie ggf. einen Namen hinzu. Beachten Sie in der Abbildung oben auch die Zeilennummerierung. Um eine Region zu löschen, löschen Sie die beiden `--region` und `--endregion` Markierungen.

Einfügen von Kommentaren in ein SQL-Script
Text in einem SQL-Script kann auskommentiert werden. Diese Script-Abschnitte werden bei der Ausführung des Scripts übersprungen.

- Um einen Block auszukommentieren, markieren Sie den Block, rechtsklicken Sie und wählen Sie den Befehl **Blockommentar einfügen/entfernen**. Um den Blockkommentar zu entfernen, rechtsklicken Sie und wählen Sie den Befehl **Blockommentar einfügen/entfernen**.
- Um eine Zeile oder den Teil einer Zeile auszukommentieren, platzieren Sie den Cursor an die Stelle, an der der Zeilenkommentar beginnen soll, rechtsklicken Sie und wählen Sie den Befehl **Zeilenkommentar einfügen/entfernen**. Um den Zeilekommentar zu entfernen, markieren Sie den Kommentar, rechtsklicken Sie und wählen Sie den Befehl **Zeilenkommentar einfügen/entfernen**

Verwendung von Lesezeichen
Lesezeichen können an bestimmten Zeilen eingefügt werden. Sie können anschließend durch die Lesezeichen im Dokument navigieren. Um ein Lesezeichen einzufügen, platzieren Sie den Cursor in die gewünschte Zeile, rechtsklicken Sie und wählen Sie den Befehl **Lesezeichen einfügen/löschen**. Um zum nächsten oder vorhergehenden Leszeichen zu navigieren, wählen Sie den Befehl **Nächstes Lesezeichen** bzw. **Vorheriges Lesezeichen**. Um ein Lesezeichen zu entfernen, platzieren Sie den Cursor in die Zeile, in der das Lesezeichen entfernt werden soll, rechtsklicken Sie und wählen Sie den Befehl **Lesezeichen einfügen/löschen.** Um alle Lesezeichen zu entfernen, rechtsklicken Sie und wählen Sie den Befehl **Alle Lesezeichen löschen.**

Abfragefenster: Arbeiten mit Abfragen

Nachdem Sie eine Verbindung zu einer Datenbank hergestellt haben, kann ein SQL-Script im SQL Editor eingegeben und ausgeführt werden. In diesem Abschnitt wird beschrieben:

- wie ein SQL-Script in SQL Editor eingegeben wird.
- wie das Script im Fenster "Datenbankabfrage" ausgeführt wird.

In diesem Abschnitt werden die folgenden Schaltflächen verwendet:

	Ausführen	Führt die aktuell ausgewählte SQL-Anweisung aus. Wenn das Script mehrere Anweisungen enthält und keine ausgewählt wird, werden alle ausgeführt.
	Zur Datenbearbeitung ausführen	Wie der Befehl "Ausführen", mit der Ausnahme, dass die Ergebnisse (auf dem Register "Ergebnisse") bearbeitet werden können.
	SQL-Datei importieren	Öffnet eine SQL-Datei im SQL Editor.

Erstellen von SQL-Anweisungen und Scripts im SQL Editor
Zum Erstellen von SQL-Anweisungen und Scripts können die folgenden GUI-Methoden verwendet werden:

- *Drag-and-Drop:* Ziehen Sie ein Objekt aus dem Browser-Fenster in den SQL Editor. Eine SQL-Anweisung wird generiert, um die Datenbank nach diesem Objekt abzufragen.
- *Kontextmenü:* Rechtsklicken Sie auf ein Objekt im Browser-Fenster und wählen Sie den Befehl **In SQL Editor anzeigen | Select**.
- *Manuelle Eingabe:* Geben Sie die SQL-Anweisung direkt im SQL Editor ein. Beim Bearbeiten hilft Ihnen die Autokomplettierungsfunktion.
- *SQL-Script importieren:* Klicken Sie in der Symbolleiste des Fensters "Datenbankabfrage" auf die Schaltfläche **SQL-Datei importieren.**

Ausführen von SQL-Anweisungen
Wenn das SQL-Script im SQL Editor mehr als eine SQL-Anweisung enthält, wählen Sie die gewünschte Anweisung aus und klicken Sie in der Symbolleiste des Fensters "Datenbankabfrage" entweder auf die Schaltfläche **Ausführen** oder auf **Zur Datenbearbeitung ausführen**. Wenn im SQL-Script keine Anweisung ausgewählt ist, werden alle Anweisungen im Script ausgeführt. Die Datenbankdaten werden abgerufen und in einem Raster auf dem Ergebnisregister angezeigt. Wenn Sie den Befehl **Zur Datenbearbeitung ausführen** gewählt haben, können die abgerufenen Daten im Ergebnisregister bearbeitet werden. Meldungen über die Ausführung der Anweisung werden auf dem Meldungsregister angezeigt.

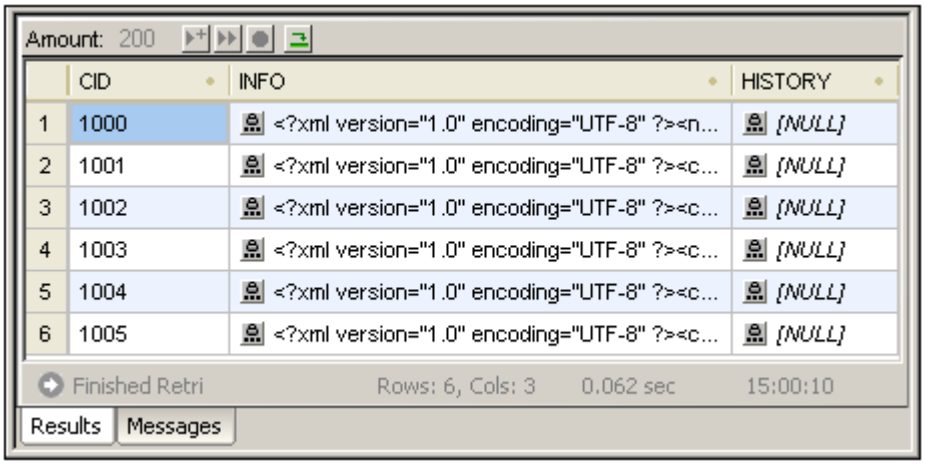
Ergebnisse und Meldungen

Das Ergebnis-/Meldungsfenster hat zwei Register:

- Auf dem Register "Ergebnisse" werden die von der Abfrage abgerufenen Daten angezeigt.
- Auf dem Register Register "Meldungen" sehen Sie Meldungen über die Ausführung der Abfrage.

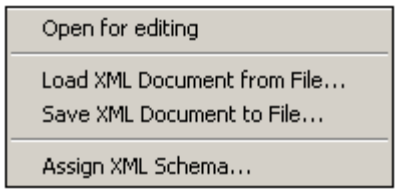
© 2007 Altova GmbH

Register "Ergebnisse"
Die von der Abfrage abgerufenen Daten werden in Tabellenform auf dem Ergebnisregister angezeigt (*Abbildung unten*).



Die folgenden Operationen können auf dem Ergebnisregister ausgeführt werden:

- *Nach einer Spalte sortieren:* Rechtsklicken Sie an eine beliebige Stelle in der Spalte, nach der die Ergebnisse sortiert werden sollen und wählen Sie den Befehl **Sortierung | Aufsteigend/Absteigend/Standardeinstellung wiederherstellen**.
- *Kopieren in die Zwischenablage:* Dieser Vorgang erfolgt in zwei Schritten: (i) Auswählen des Datenbereichs; und (ii) Kopieren der Auswahl. Die Daten können auf verschiedene Arten ausgewählt werden: (i) durch Klicken auf eine Spaltenüberschrift zur Auswahl der Spalte bzw. Zeile; (ii) durch Auswählen einzelner Zellen (Zur Auswahl mehrerer Zellen verwenden Sie **Umschalt** und/oder **Strg**); (iii) durch Rechtsklick auf eine Zelle und Auswahl von **Auswahl | Zeile/Spalte/Alle**. Klicken Sie anschließend auf die rechte Maustaste und wählen Sie den Befehl **Ausgewählte Zellen kopieren.** Daraufhin wird die Auswahl in die Zwischenablage kopiert, von wo sie in andere Applikationen eingefügt werden kann.
- *Bearbeiten von Datensätzen:* Wenn die Abfrage mit dem Befehl **Zur Datenbearbeitung ausführen** ausgeführt wurde, können einzelne Felder bearbeitet werden. Um die Änderungen zu übernehmen, klicken Sie in der Symbolleiste des Ergebnisregisters auf die Schaltfläche "Übernehmen".
- *Bearbeiten von XML-Datensätzen:* Wenn es sich bei einem editierbaren Feld um ein XML-Feld handelt (derzeit werden nur IBM DB2 XML-Datenbanken unterstützt), wird bei Klick auf das XML-Symbol im Ergebnisraster das Menü"XML bearbeiten" geöffnet (*Abbildung unten*)



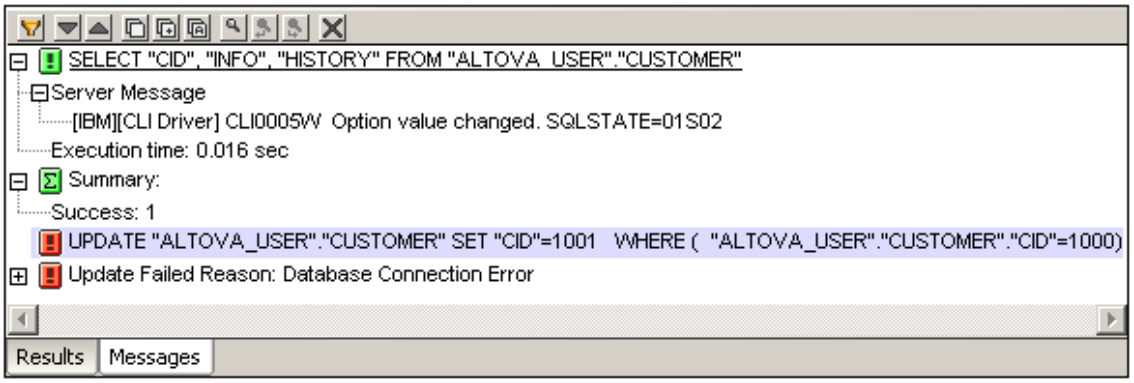
Mit dem Befehl **Zur Bearbeitung öffnen** wird das XML-Dokument in einem XMLSpy Fenster geöffnet. Wenn dieses Dokument gespeichert wird und in der Datenbankabfrage-Symbolleiste die Schaltfläche "XML-Änderungen automatisch übernehmen" beim Öffnen des Dokuments aktiviert war, werden die Änderungen am XML-Dokument in der Datenbank übernommen. (Beachten Sie: Um zwischen dem

XML-Dokumentfenster und dem Fenster "Datenbankabfrage" zu wechseln, müssen Sie den Befehl **DB | Datenbank abfragen** aufrufen). Mit dem Befehl **XML-Dokument aus Datei laden** wird ein externes XML-Dokument in das ausgewählte Feld in der Datenbank geladen. Mit dem Befehl **XML-Dokument in Datei speichern** wird das XML-Dokument im ausgewählten Datenbankfeld unter einem Dateipfad Ihrer Wahl gespeichert. Mit dem Befehl "XML-Schema zuweisen" wird das Dialogfeld "XML-Schema auswählen" aufgerufen, in dem Sie ein XML-Schema auswählen können, das Sie dem XML-Dokument zuweisen. Die Zuweisung wird in der Datenbank gespeichert. Die Zuweisung von XML-Schemas wird im Abschnitt IBM DB2 | Zuweisen eines XML-Schemas näher behandelt.

Das Ergebnisregister hat die folgenden Symbolleistenschaltflächen:

	Gehe zu Anweisung	Markiert die SQL Anweisung im SQL Editor, die das entsprechende Ergebnis erzeugt hat.
	Neue Zeile hinzufügen	Fügt eine neue Zeile zum Ergebnisraster hinzu.
	Zeile löschen	Löscht die aktuelle Zeile im Ergebnisraster.
	Änderungen am Ergebnisraster rückgängig machen	Macht alle Änderungen am Ergebnisraster rückgängig.
Commit	**Übernehmen**	Übernimmt die im Ergebnisraster vorgenommenen Änderungen in der Datenbank.

Register "Meldungen"
Das Register "Meldungen" enthält Informationen über die zuvor ausgeführte SQL-Anweisung und gibt Fehlermeldungen und Warnmeldungen aus.



Die Symbolleiste des Meldungsregisters enthält Schaltflächen, mit denen Sie die Ansicht anpassen, darin navigieren sowie Meldungen in die Zwischenablage kopieren können. Mit Hilfe der Schaltfläche **Filter** können Sie bestimmte Arten von Meldungen ein- und ausblenden. Mit Hilfe der Schaltflächen **Nächste** und **Vorherige** wird die jeweils vorige bzw. nächste Meldung in der Liste markiert. Meldungen können auch mit oder ohne Ihre Sub-Einträge in die Zwischenablage kopiert werden, sodass sie in Dokumente eingefügt werden können. Mit Hilfe der Funktion **Suchen** können Sie einen Suchbegriff angeben und dann die Liste nach diesem Begriff durchsuchen. Mit der Schaltfläche **Löschen** können Sie den Inhalt des Berichtsfensters löschen.

© 2007 Altova GmbH

Anmerkung: Diese Symbolleisten-Befehle stehen auch im Kontextmenü zur Verfügung.

15.9.3 IBM DB2

Das Menü **IBM DB2** enthält ein Untermenü mit den folgenden Befehlen (i) zum Registrieren und Löschen der Registrierung von Schemas bei einer IBM DB2 Datenbank (**Verwalten von XML-Schemas**), und (ii) zum Zuweisen von Schemas für die XML-Dateivalidierung (**Zuweisen eines XML-Schemas**).

Für beide Funktionen muss eine Verbindung zur jeweiligen IBM DB2-Datenbank bestehen. Wie man eine Verbindung zur Datenbank herstellt, ist im Abschnitt Herstellen einer Verbindung zur Datenquellen beschrieben. In diesem Abschnitt wird beschrieben, wie Schemas in einer IBM DB2 Datenbank verwaltet und wie XML-Schemas einer Datenbank-XML-Datei zugewiesen werden.

Anmerkung: Das Ergebnisraster des Fensters "Datenbankabfrage" enthält wichtige Funktionen für die Arbeit mit XML-Dateien in IBM DB2-Datenbanken. Dazu gehören Funktionen zum Bearbeiten, Laden von XML-Dateien in Datenbank-XML-Dateien, externen Speichern von DB-XML-Dateien und Zuweisen von XML-Schemas zu DB-XML-Dateien.

XML-Schemas verwalten

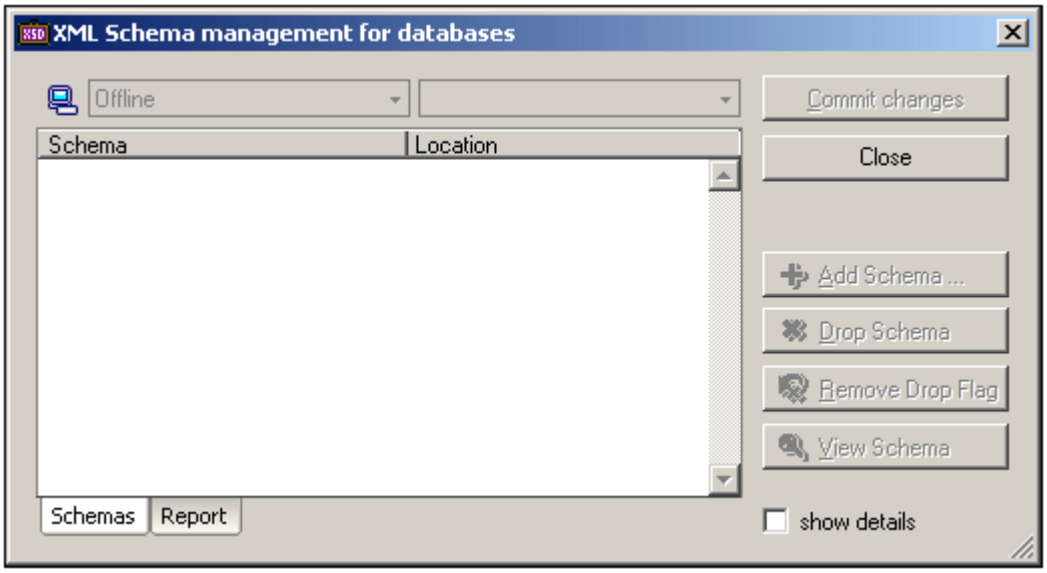
Mit dem Befehl **XML-Schemas verwalten** können Schemas aus einzelnen Datenbank-Schemas in einer IBM DB2-Datenbank hinzugefügt und gelöscht werden. Dazu müssen folgendes tun:

- eine Verbindung zu einer IBM DB2-Datenbank herstellen
- das/die Datenbank-Schema(s) auswählen, für das/die XML-Schemas hinzugefügt oder gelöscht werden sollen und
- anschließend die Schemaverwaltungsaktionen durchführen.

Diese Schritte sind im Folgenden beschrieben.

Herstellen einer Verbindung zur IBM DB2-Datenbank

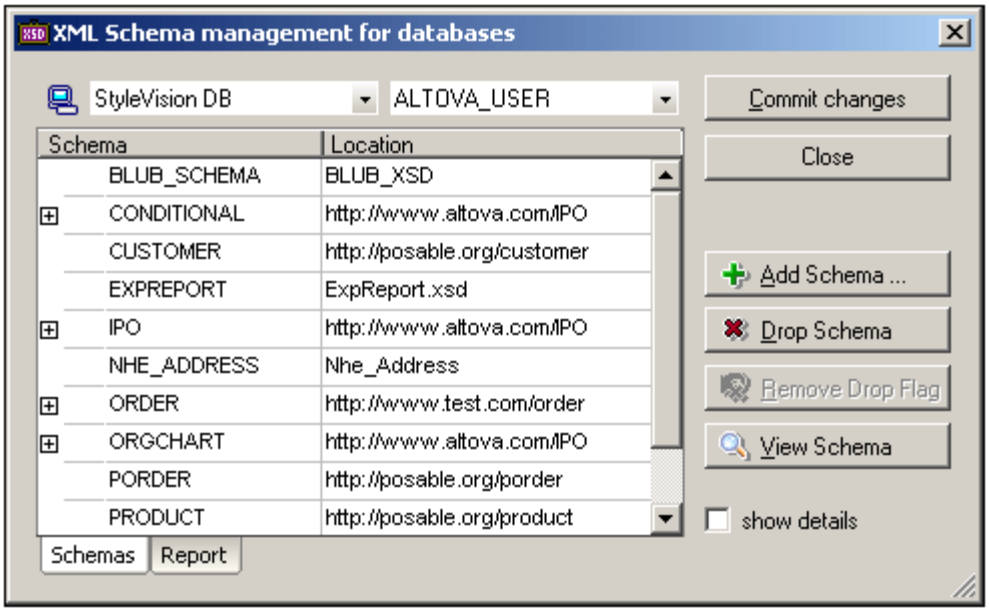
Wenn Sie den Befehl **XML-Schemas verwalten** auswählen, wird das Dialogfeld "XML-Schemaverwaltung für Datenbanken" aufgerufen (*Abbildung unten*).



Um die Schemas einer IBM DB2-Datenbank zu verwalten, müssen Sie zuerst eine Verbindung zur Datenbank herstellen. Klicken Sie dazu im Dialogfeld auf die Schaltfläche "Verbindungsassistent" . Daraufhin wird das gleichnamige Dialogfeld aufgerufen, über das Sie die Verbindung zur Datenbank herstellen können. Die Verwendung des Verbindungsassistenten ist im Abschnitt Menü "DB" | Herstellen einer Verbindung zu einer Datenquelle beschrieben.

Anzeigen einer Liste von XML-Schemas

Nachdem die Verbindung zur IBM DB2-Datenbank hergestellt wurde, wird die Datenbank in der Auswahlliste auf der linken Seite (*siehe Abbildung unten*) aufgelistet, Wenn derzeit mehrere Verbindungen bestehen, können Sie die gewünschte Datenbank in der Auswahlliste auswählen. In der Abbildung unten ist die Datenbank "StyleVision DB" ausgewählt.



In der Auswahlliste auf der rechten Seite sind alle Datenbankschemas der ausgewählten IBM DB2-Datenbank aufgelistet. Wenn ein Datenbankschema in dieser Auswahlliste ausgewählt ist,

werden alle für das ausgewählte Datenbankschema registrierten XML-Schemas im Hauptfenster angezeigt. In der Abbildung oben werden alle mit dem Datenbankschema `Altova_User` registrierten Schemas zusammen mit Ihren Pfaden aufgelistet. Wenn Sie das Kontrollkästchen "Details anzeigen" aktivieren, werden im Hauptfenster weitere Datenspalten angezeigt.

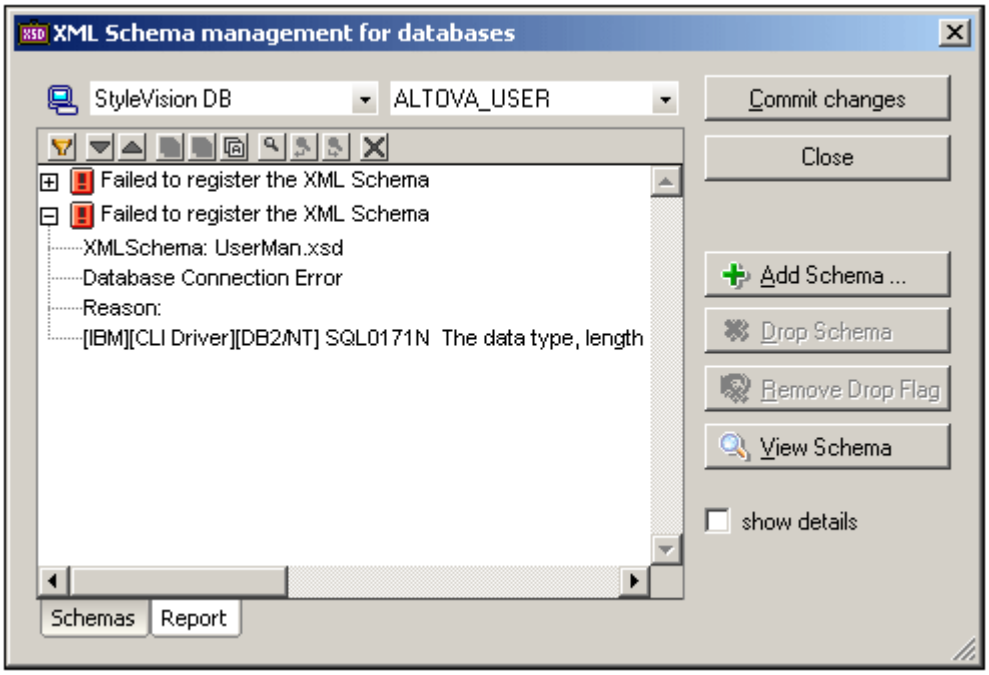
Verwalten der XML-Schemas
Die Liste der Schemas im Hauptfenster enthält die für das ausgewählte Datenbankschema registrierten Schemas. Sie können nach Anzeige der Liste der XML-Schemas Schemas zur Liste hinzufügen oder aus der Liste löschen.

Um ein Schema hinzuzufügen, klicken Sie auf die Schaltfläche **Hinzufügen,** suchen Sie nach der gewünschten Schemadatei und wählen Sie sie aus. Die ausgewählte Schemadatei wird zur Liste im Hauptfenster hinzugefügt. Wenn Sie auf die Schaltfläche **Änderungen übernehmen** klicken, werden die neu hinzugefügten Schemas im Datenbankschema registriert.

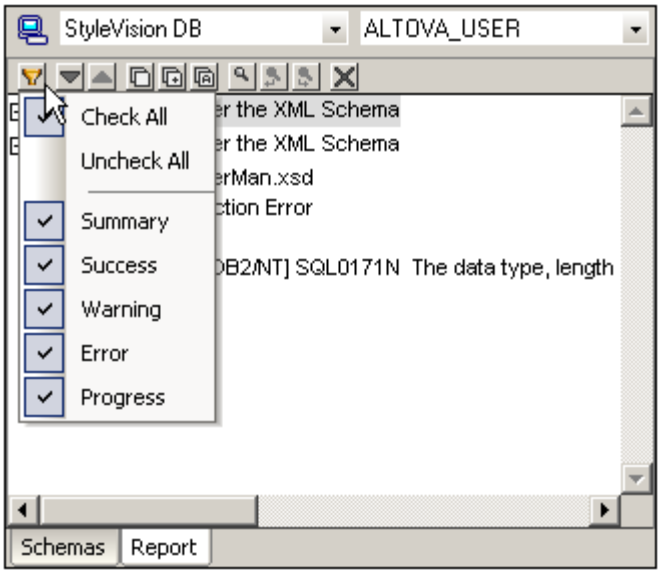
Um ein Schema zu löschen, wählen Sie das Schema im Hauptfenster aus und klicken Sie auf die Schaltfläche **Schema löschen.** Dem Schema wird ein Drop Flag zugewiesen, um anzuzeigen, dass es bei der Übernahme der nächsten Änderungen gelöscht wird. Sie können das Drop Flag löschen, indem Sie das so gekennzeichnete Schema auswählen und auf die Schaltfläche **Drop Flag entfernen** klicken. Wenn Sie auf die Schaltfläche **Änderungen übernehmen** klicken, wird die Registrierung aller für die Löschung markierten Schemas im Datenbankschema aufgehoben.

Wenn Sie auf die Schaltfläche "Schema anzeigen" klicken, wird das Schema in XMLSpy geöffnet. Um das Dialogfeld "XML-Schemaverwaltung" zu schließen, klicken Sie auf die Schaltfläche **Schließen.**

Berichte
Wenn Sie auf die Schaltfläche **Änderungen übernehmen** klicken, werden die von Ihnen vorgenommenen Änderungen in der Datenbank übernommen. Ein Bericht der Aktion "Übernehmen" wird im Berichtsfenster (*Abbildung unten*) angezeigt, sodass Sie sehen, ob die Aktion erfolgreich durchgeführt wurde, und um mögliche Fehler beheben zu können. Anschließend wird ein weiterer Bericht unterhalb des vorherigen angezeigt.



Das Berichtsfenster verfügt über Schaltflächen, mit denen Sie die Anzeige der Berichtsliste anpassen, in der Liste navigieren, Berichtsmeldungen kopieren, nach Text suchen und den Inhalt des Fensters löschen können (*siehe Abbildung unten*).



Mit Hilfe der Schaltfläche **Filter** können Sie bestimmte Arten von Meldungen ein oder ausblenden. Mit Hilfe der Schaltflächen **Nächste** und **Vorherige** wird die jeweils vorige bzw. nächste Meldung in der Liste markiert. Meldungen können auch mit oder ohne Ihre Sub-Einträge in die Zwischenablage kopiert werden, sodass sie in Dokumente eingefügt werden können. Mit Hilfe der Funktion **Suchen** können Sie einen Suchbegriff angeben und dann die Liste nach diesem Begriff durchsuchen. Mit der Schaltfläche **Löschen** können Sie den Inhalt des Berichtsfensters löschen.

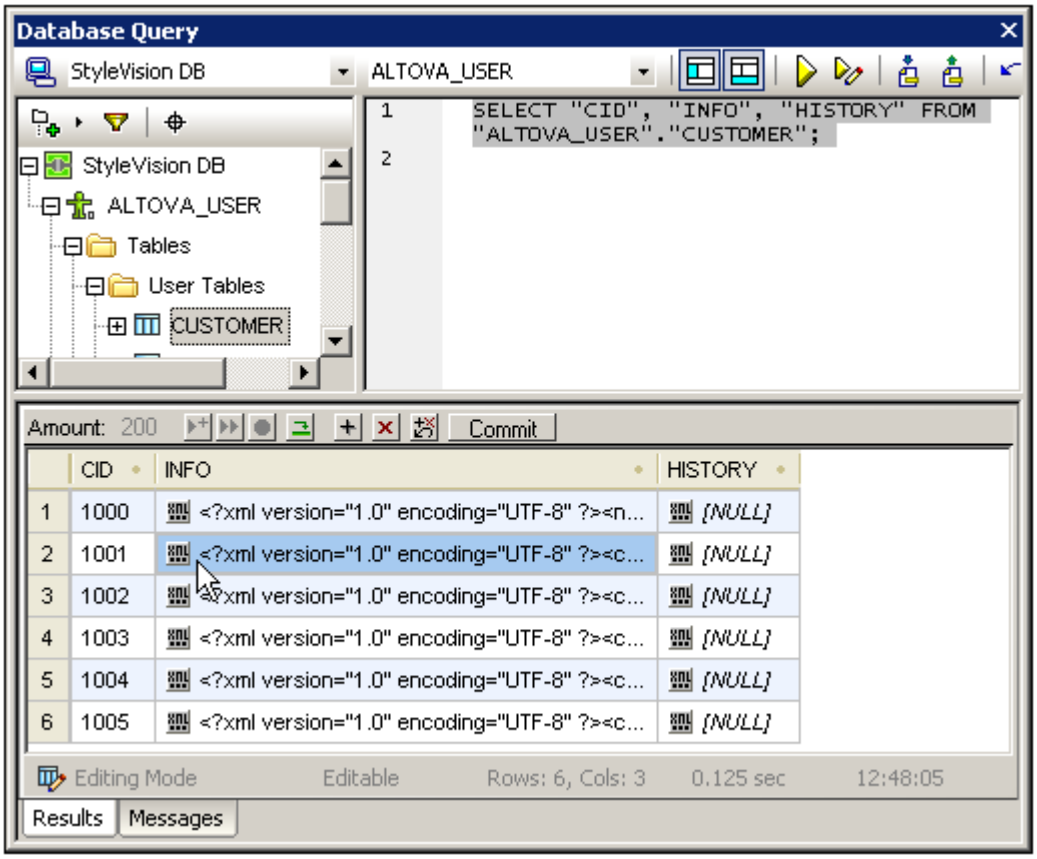
© 2007 Altova GmbH

XML-Schema zuweisen

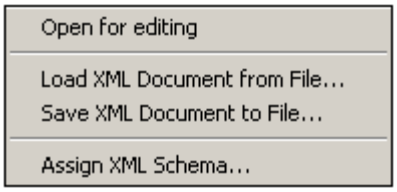
Mit dem Befehl "XML-Schema zuweisen" wird einer XML-Datei, die zur Bearbeitung über das Ergebnisraster des Fensters "Datenbankabfrage" geöffnet wurde, ein XML-Schema zugewiesen. Nach Zuweisung des Schemas kann die XML-Datei gegen das zugewiesene Schema validiert werden. Die Zuweisung wird in die Datenbank geschrieben, wenn die XML-Datei in XMLSpy gespeichert wurde.

Öffnen einer Datenbank-XML-Datei zur Bearbeitung

Wenn im Fenster "Datenbankabfrage" eine Abfrage an einer XML-Datenbank durchgeführt werden soll und die Abfrage zur Bearbeitung von Daten ausgeführt wird, haben Sie im Ergebnisraster im unteren Bereich des Fensters "Datenbankabfrage" Zugriff auf die XML-Dateien in der Datenbank, sodass diese bearbeitet werden können (*siehe Abbildung unten*).

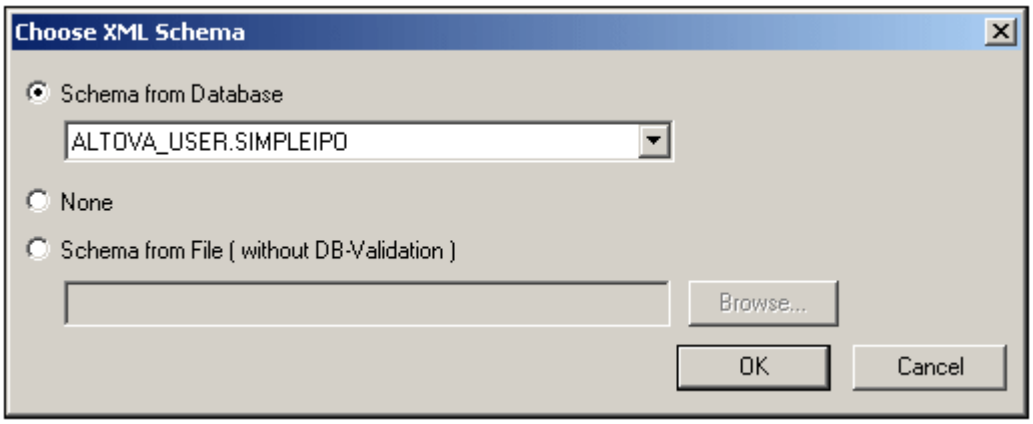


Wenn Sie auf die Schaltfläche "XML" klicken, erscheint das folgenden Menü.



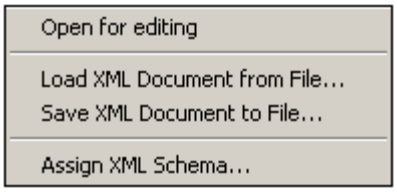
Wenn Sie den Befehl **Zur Bearbeitung öffnen** auswählen, wird das XML-Dokument in XMLSpy geöffnet, wo es bearbeitet werden kann.

Zuweisen eines Schemas zur Datenbank-XML-Datei
Der Befehl **IBM DB2 | XML-Schema zuweisen** steht zur Verfügung, wenn die Datenbank-XML-Datei zur Bearbeitung in XMLSpy geöffnet wird. Wenn Sie den Befehl **XML-Schema zuweisen** aufrufen, während das XML-Dokument in XMLSpy aktiv ist, wird das Dialogfeld "XML-Schema auswählen" (*Abbildung unten*) geöffnet.



Sie können ein Schema aus den in der Datenbank gespeicherten Schemas auswählen (diese Schemas werden in der Dropdown-Liste der Auswahlliste "Datenbank" aufgelistet) oder aus externen Dateien, die durchsucht werden können. Wenn Sie auf **OK** klicken, wird das Schema der XML-Datei zugewiesen. Beachten Sie, dass die Zuweisung nicht in die XML-Datei geschrieben wird. Die Schemazuweisung wird in der Datenbank gespeichert, wenn das Dokument in XMLSpy gespeichert wird - und falls in der Datenbankabfrage-Symbolleiste beim Öffnen des Dokuments die Schaltfläche "XML-Änderungen automatisch übernehmen" aktiviert war. Beachten Sie, dass die Schemazuweisung in die Datenbank und nicht in die XML-Datei geschrieben wird.

Anmerkung: Das Menü "XML bearbeiten" im Ergebnisraster des Fensters "Datenbankabfrage" hat ebenfalls einen Befehl **XML-Schema zuweisen** (*siehe Abbildung unten*), der ebenfalls einer Datenbank-XML-Datei ein Schema zuweist.



Der Unterschied zwischen den beiden "XML-Schema zuweisen"-Befehlen besteht darin, dass Sie mit Hilfe des Befehls im Menü **DB | IBM DB2** ein XML-Schema zuweisen können, während Sie die XML-Datei bearbeiten, sodass Sie die Schemazuweisungen während der Editierung des XML-Dokuments ändern und das XML-Dokument sofort validieren können.

15.9.4 Tamino

Software AG und Altova Inc. haben ein weltweit gültiges Abkommen abgeschlossen, mit dem User von XMLSpy einfachen und günstigen Zugang zu Tamino erhalten. Mit dieser Partnerschaft erreicht Software AG die ständig wachsende Gemeinschaft von Developern, die Produkte von Altova verwenden.

Dank dieses Abkommens erhalten User von Altova's XMLSpy und Software AG's Tamino ein

integriertes Produktbündel. Tamino XML Server steht auf multiplen Plattformen von PC bis Mainframe zur Verfügung.

XMLSpy und Tamino (Limited Edition) erhalten Sie als Download und zum Erwerb über unsere Bestellseite im Web. Die Tamino Datenbankkapazität liegt je nach Anzahl der erworbenen Lizenzen für XMLSpy zwischen 50MB (für XMLSpy Enterprise/Professional Einzellizenz) bis 1000 MB.

Nach erfolgtem Zahlungseingang erhalten Sie eine Email mit Angaben zum Download von Tamino.

Installation von Tamino

Nachdem Sie das Tamino und XMLSpy Bündel heruntergeladen haben, können Sie mit der Installation beginnen.

Allgemeine Installationsschritte:

1. Download und Installation von XMLSpy
2. Download von Tamino XML Server
3. Installation von Apache Server (falls noch nicht installiert)
4. Installation von Tamino Server 3.1.1
5. Erzeugen einer Datenbank in Tamino
6. XMLSpy verwenden

Spezifische Tamino-Installationsanweisungen liegen im HTML-Format vor. Öffnen Sie hierzu die Tamino Dokumentation unter:

<Root>/Windows/INO/Docu/overview.htm

"<**Root**>" steht für das Verzeichnis, in welches Sie Tamino entpackt haben. Wählen Sie den Eintrag "Installing Tamino" (Tamino installieren).

Umfang der vorliegenden Dokumentation:
Es wird davon ausgegangen, dass die Installation von XMLSpy Enterprise oder Professional Edition, sowie Tamino Server 3.1.1. erfolgreich durchgeführt wurde.

Installation von WebDAV

Installation und Verwendung von WebDAV:
XMLSpy unterstützt WebDAV. Die Installation von WebDAV ist optional. Die Tamino Funktionalitäten für XMLSpy sind aufgrund von Synchronisationseinschränkungen nicht mit WebDAV kompatibel.

Installation von WebDAV:

- Installieren Sie WebDAV Server von Software AG.
- Erstellen Sie einen WebDAV Speicher, und starten Sie den WebDAV Server erneut.

Tamino Schema-Extensions

Während XML Schema alle logischen Aspekte der Dokumenttyp-Definition (DTD) berücksichtigt, wird die Definition physischer Aspekte nicht behandelt. Viele XML Prozessoren benötigen deshalb zusätzliche Informationen beim Verarbeiten bestimmter Klassen von Dokumenten.

Tamino benötigt beispielsweise eine Fülle weiterer Informationen über die Dokumentsammlung

(Datensammlung), der ein bestimmtes Dokument angehört, welche Elemente für die Indizierung verwendet werden sollen, welche Indizierung verwendet werden soll, wie Elemente eines Dokuments auf Felder in externen Datenbanken gemappt werden sollen, usw...

Hierfür bietet XML Schema einen Extensionsmechanismus. Jedes Schema, Element oder Attribut kann mit einer oder mehreren Annotationen versehen werden.

Jede **Annotation** kann aus zwei Child Elementen bestehen: **documentation** und **appinfo**. Das Element documentation enthält lesbare Dokumentation, während appinfo Informationen zur maschinellen Verarbeitung enthält.

Tamino verwendet **appinfo** zum Speichern von Tamino-bezogenen Informationen in einer Schemadatei:

```
<?xml version = "1.0" encoding = "UTF-8"?>
<xs:schema xmlns:xs = "http://www.w3.org/2001/XMLSchema"
        xmlns:tsd =
   "http://namespaces.softwareag.com/tamino/TaminoSchemaDefinition">
  <xs:annotation>
    <xs:appinfo>
      <tsd:schemaInfo name = "album">
        <tsd:collection name = "encyclopedia"/>
        <tsd:doctype name = "album">
          <tsd:logical>
            <tsd:content>open</tsd:content>
            <tsd:accessOptions>
              <tsd:read/>
              <tsd:insert/>
              <tsd:delete/>
              <tsd:update/>
            </tsd:accessOptions>
          </tsd:logical>
        </tsd:doctype>xsd:schemaInfo>
    </xs:appinfo>
  </xs:annotation>
...
</xs:schema>
```

Diese Annotationen werden vom Tamino Schema Editor erzeugt, sowie in der XMLSpy Schema Design Ansicht, wenn Sie mit Tamino Erweiterungen arbeiten.

Erstellen, Speichern und Auflisten von Tamino Schemas

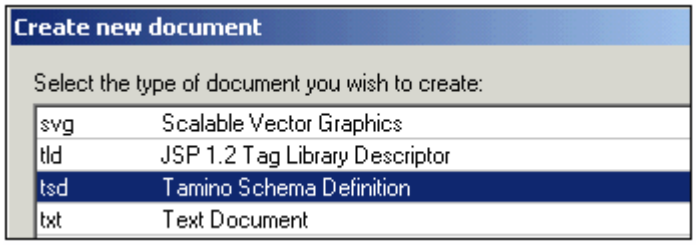
Jede Kommunikation zwischen Tamino und XMLSpy erfolgt über das HTTP Kommunikationsprotokoll.

Tamino Datenbank befüllen
Um Daten in Tamino speichern zu können, ist das Erstellen und Speichern eines Schemas in Tamino nötig. Im Weiteren wird ein einfaches Schema erzeugt. Dieses Schema finden Sie auch im Verzeichnis Examples unter dem Namen **BMW.tsd**.

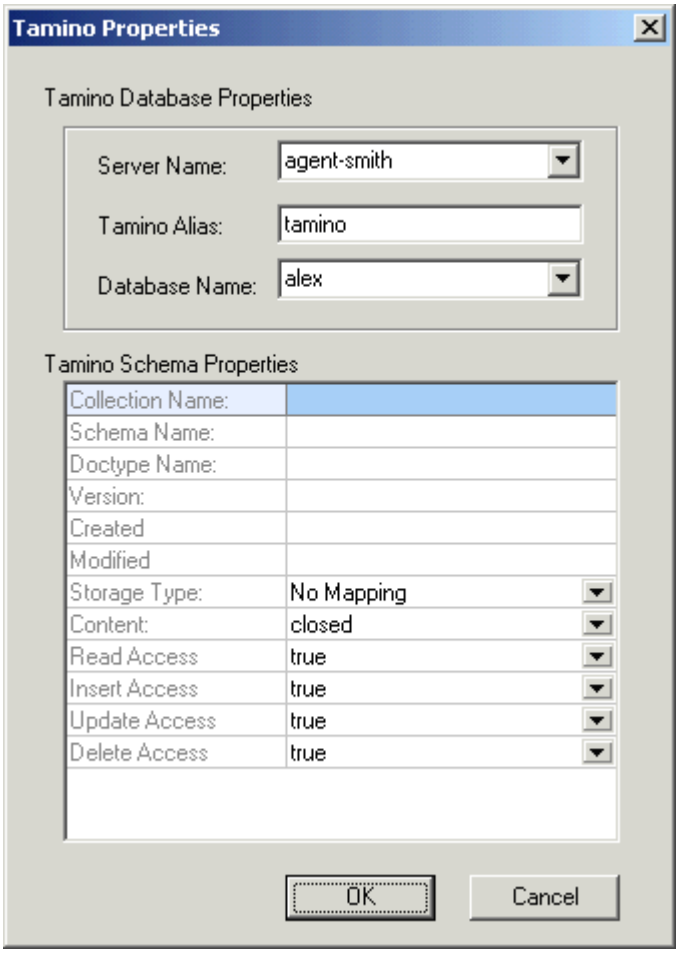
So erstellen Sie eine Tamino Schema Definitionsdatei:

1. Erstellen Sie ein Schema über die Menüoption **Datei | Neu** in XMLSpy, und wählen Sie den Eintrag **tsd Tamino Schema Definition**.

 © 2007 Altova GmbH



Das Dialogfeld Tamino Eigenschaften wird geöffnet. Hier definieren Sie die Eigenschaften der Datenbank.



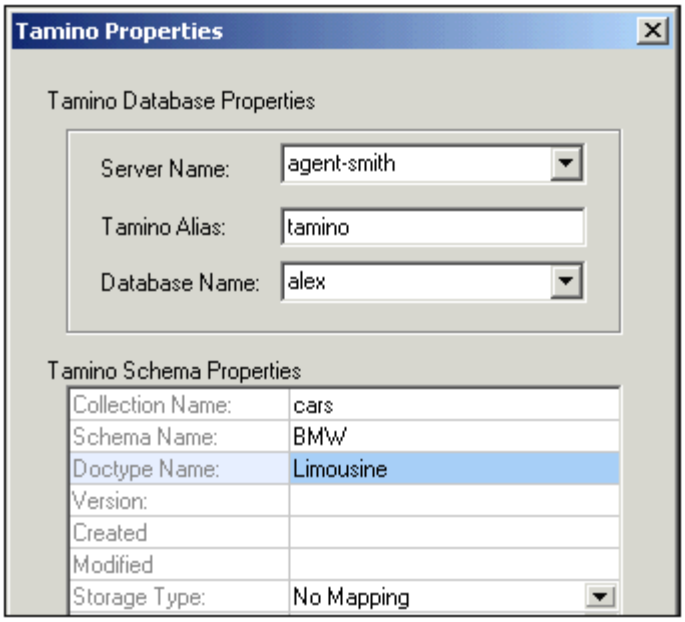
Falls noch nicht angegeben, geben Sie die Namen für Server, Datenbank und Tamino Alias in die entsprechenden Felder ein.

2. Geben Sie den Namen der Datensammlung, des Schemas und des Doctypes in die entsprechenden Felder ein, und bestätigen Sie mit OK. Die hier angegebenen Informationen werden in Tamino übernommen.

Datensammlung Name:	cars
Schema Name:	BMW
Doctype Name:	Limousine

Bitte beachten Sie:

Der Tamino Alias wird vom Datenbank-Administrator aus Sicherheitszwecken vergeben, oder wenn ein Alias bereits besetzt ist. Bitte wenden Sie sich an Ihren Datenbank-Administrator für weitere Informationen zu Tamino Alias.



Eine Tamino Schema Definitionsdatei wird erzeugt, die in der Schema Design Ansicht dargestellt wird.

3. Geben Sie den Doctype Name "Limousine" im Root Element-Feld ein. (Der Doctype Name kann ein beliebiges globales Element sein, und muss nicht unbedingt das Root Element sein. Sie benötigen diesen Namen allerdings in der Schema-Datei zum Speichern in Tamino.)

4. Jetzt ist die Schemadatei mit der Tamino Datenbank kompatibel. In der Eingabehilfe Details wurde ein Register mit Namen "Tamino" angehängt. Tamino Extensionen können über die Eingabehilfe Details für Elemente und Attribute zugewiesen werden. Durch Klicken auf das Register Text-Ansicht wird die Schemadefinition wie folgt dargestellt:

© 2007 Altova GmbH

```
<xs:schema
xmlns:tsd="http://namespaces.softwareag.com/tamino/TaminoSchemaDefiniti
on" xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified">
    <xs:annotation>
        <xs:appinfo>
            <tsd:schemaInfo name="BMW">
                <tsd:collection name="cars"/>
                <tsd:doctype name="Limousine">
                    <tsd:logical>
                        <tsd:content>closed</tsd:content>
                    </tsd:logical>
                </tsd:doctype>
            </tsd:schemaInfo>
        </xs:appinfo>
    </xs:annotation>
    <xs:element name="Limousine">
        <xs:annotation>
            <xs:documentation>The root element is usually named the same
as the doctype.</xs:documentation>
        </xs:annotation>
    </xs:element>
</xs:schema>
```

5. Entwerfen Sie Ihr Schema mit dem Schemaeditor wie folgt:

   Hängen Sie am Root-Element "Limousine" einen Sequenz-Node an, und danach folgende Child Element Nodes:

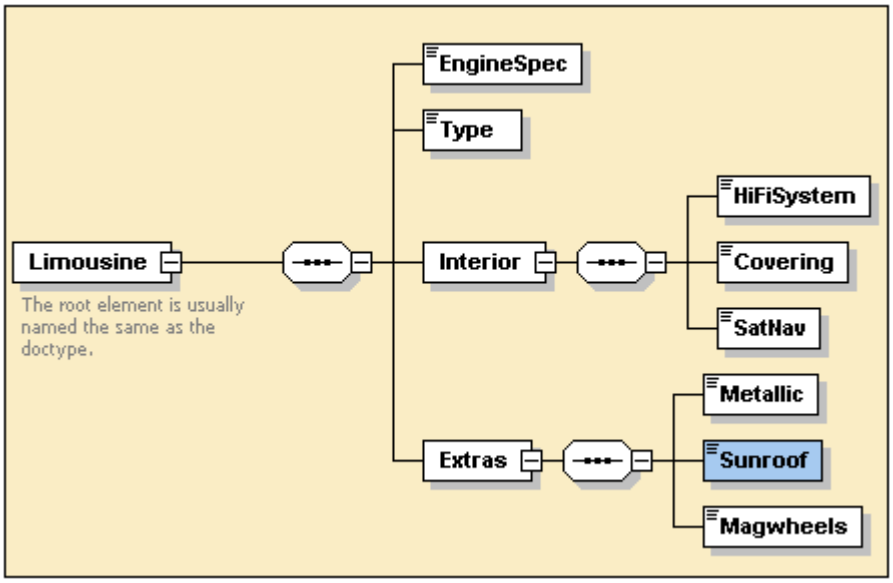
EngineSpec	xs:string
Type	xs:string
Interior	
Extras	

   Hängen Sie einen Sequenz-Node an das Element "Interior" an, und dann folgende Children:

HiFiSystem	xs:string
Covering	xs:string
SatNav	xs:string

   Hängen Sie einen Sequenz-Node an das Element "Extras" an, und dann folgende Children:

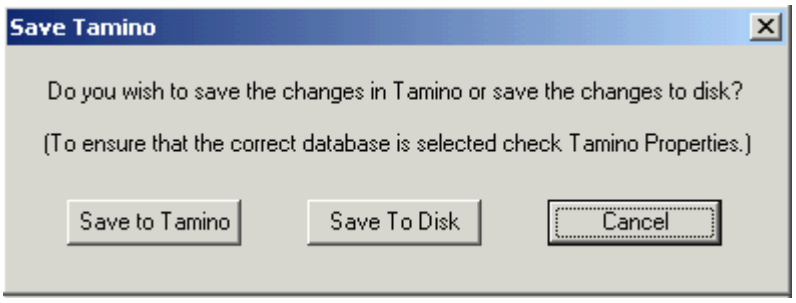
Metallic	xs:boolean
Sunroof	xs:boolean
Magwheels	xs:boolean



Bitte beachten Sie:
Tamino unterstützt keine **Global complex** types!

So speichern Sie ein Tamino Schema:

1. Verwenden Sie die Tastenkombination Strg+S, und klicken Sie dann auf die Schaltfläche "In Tamino speichern".
   Das Schema wird in der Tamino Datenbank gespeichert. Sie können auch die Menüoption **Konvertieren | Tamino | In Tamino speichern...** wählen, um das Schema direkt in der Datenbank zu speichern.



Bitte beachten Sie:
Wenn Sie ein Schema in Tamino speichern, wird automatisch der **Tamino Schema Validator** aufgerufen. Dieser Validator überprüft Tamino-bezogene Informationen wie Extensionen und spezifische Regeln, und ist nicht mit dem Validator in XMLSpy zu verwechseln. Bei Tamino Validierungsfehlern erhalten Sie eine entsprechende Fehlermeldung. Weitere Informationen zu diesem Thema erhalten Sie in der Tamino Dokumentation.

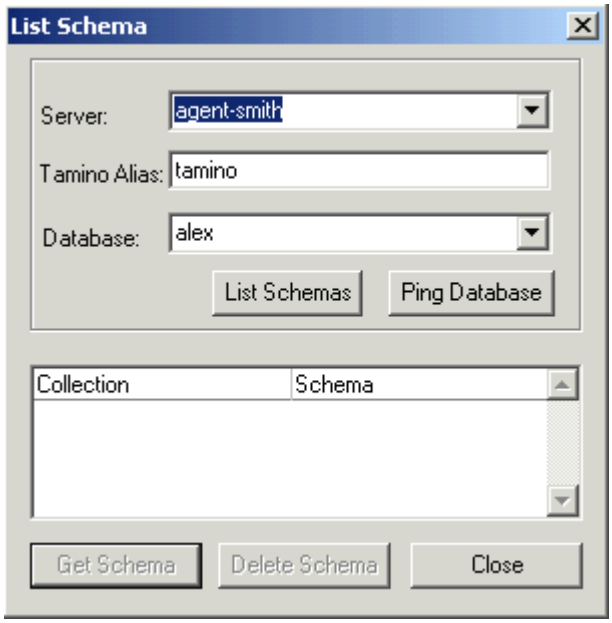
Auf Festplatte speichern
Mit der oben genannten Methode wird das Dokument in der Datenbank gespeichert, ohne es auf der Festplatte zu speichern. Im Hauptfenster wird als Name des Dokuments **Untitledxx.xsd** angezeigt. Um die Datei lokal zu speichern, verwenden Sie die Schaltfläche **"Auf Festplatte speichern"**.

Wenn Sie XML Dateien in XMLSpy erzeugen möchten, die auf Schemas basieren, die in Tamino gespeichert wurden, verwenden Sie immer den Befehl "Auf Festplatte speichern", sodass eine Kopie des Schemas lokal vorhanden ist.

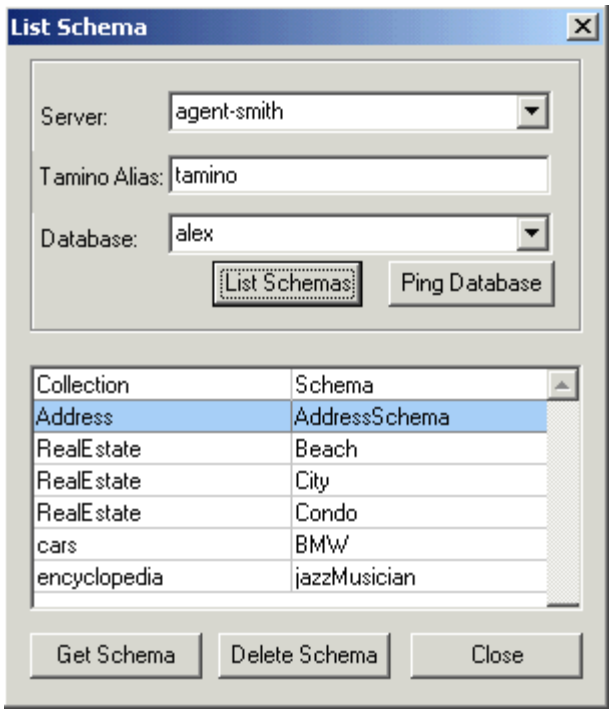
© 2007 Altova GmbH

So können Sie Schemas in der Tamino Datenbank auflisten:

1. Wählen Sie die Menüoption **DB | Tamino | Schema auflisten...**
   Das Dialogfeld "Schema auflisten" wird geöffnet.



2. Falls noch nicht angegeben, geben Sie Server- und Datenbanknamen in den entsprechenden Feldern ein.
3. Klicken Sie auf die Schaltfläche **Auflisten**.
   Eine Liste mit Schemas und verknüpften Datensammlungen der aktuellen Datenbank wird angezeigt.



Anmerkung: Über die Schaltfläche "Ping Database" können Sie die Verbindung zu

Tamino überprüfen.

So öffnen Sie ein Schema:

- Doppelklicken Sie auf den Schema-Namen im Listenfeld, oder
- Markieren Sie den entsprechenden Eintrag, und klicken Sie auf "Schema holen".

So löschen Sie ein Schema aus Tamino:

1. Zeigen Sie die Schemas im Dialogfeld "Schema auflisten".
2. Markieren Sie das zu löschende Schema, und klicken Sie auf "Schema löschen".

Validierung von Tamino Schemas

- Nachdem Sie ein Schema aus Tamino "geholt" haben, wählen Sie die Menüoption **XML | Validieren**, oder betätigen Sie die Funktionstaste F8.
  In diesem Fall wird der Validator von XMLSpy verwendet. XMLSpy Fehlermeldungen werden bei ungültigen Schemas angezeigt.

Wenn Sie ein Schema **in Tamino speichern**, wird automatisch der **Tamino Schema Validator** aufgerufen. Dieser Validator überprüft Tamino-bezogene Informationen wie Extensionen und spezifische Regeln, und ist nicht mit dem Validator in XMLSpy zu verwechseln. Bei Tamino Validierungsfehlern erhalten Sie eine entsprechende Fehlermeldung. Weitere Informationen zu diesem Thema erhalten Sie in der Tamino Dokumentation.

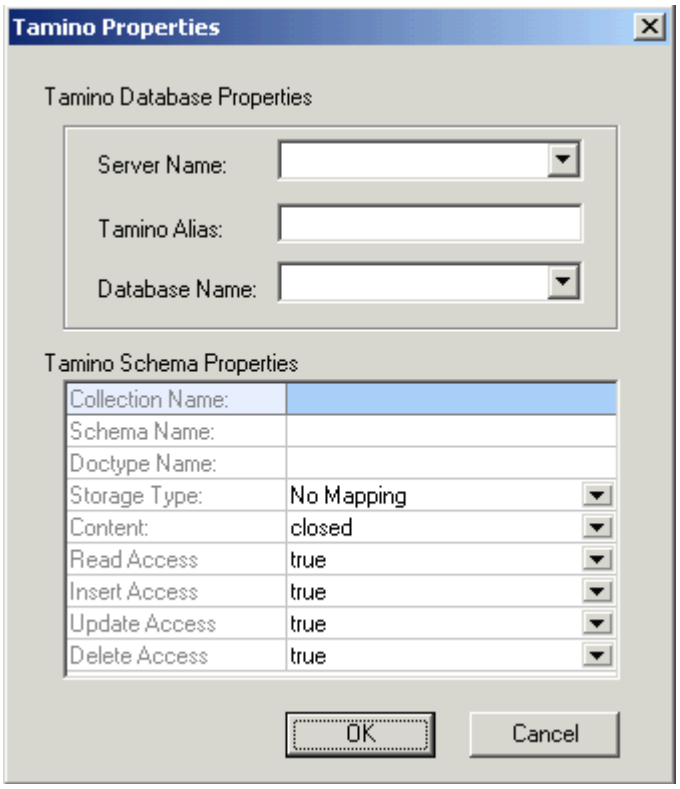
Legacy Schemas in Tamino hinzufügen

Schemas, die in XMLSpy erzeugt wurden, können auf einfache Weise in Tamino Server hinzugefügt werden.

So fügen Sie ein existierendes Schema in Tamino Server hinzu:

1. Öffnen Sie das Schema in der Schema/WSDL Design-Ansicht.
2. Wählen Sie die Menüoption **Schema-Design | Tamino Schema-Extensions aktivieren**.
   Wenn Sie ein anderes Präfix als 'xs' verwenden, wird ein Dialogfeld angezeigt. In Tamino muss das Präfix immer **'xs'** sein.
3. Klicken Sie auf OK, falls das Dialogfeld angezeigt wird.
   Ein weiteres Dialogfeld wird angezeigt, und Sie werden gefragt, ob die Dateierweiterung in .tsd geändert werden soll. Dies ist optional - zum Bestätigen klicken Sie bitte auf OK.
4. Das Dialogfeld Tamino Eigenschaften wird angezeigt. Geben Sie die Eigenschaften der Datenbank sowie die des Schemas an: Datensammlung, Schema und Doctype, und bestätigen Sie mit OK.

© 2007 Altova GmbH



5. Wählen Sie die Menüoption **DB | Tamino | In Tamino speichern**.
   Sie erhaltern eine Bestätigungsmeldung, sobald das Schema erfolgreich gespeichert wurde.

Einschränkungen für Tamino Schema Definitionen

Einschränkungen der Tamino Schema Definition Language-Implementierung (Copyright Software AG 2001)
Die in der aktuellen Version von Tamino implementierte Tamino Schema Definition Language weist die folgenden Einschränkungen auf:

Logische TSD3-Einschränkungen, die nicht durch eine Syntax-Untermenge eines logischen W3C XS Meta-Schemas abgeleitet werden können:

- Die Element-Namen jedes einzelnen lokalen Elements und aller referenzierten globalen Elemente müssen für jede Model Group (d.h. den Inhalt von 1 complexType) eindeutig sein. Das bedeutet, dass für jeden Element-Namen entweder ein lokales Element oder mehrere Referenzen auf globale Elemente zulässig sind.
- Für das Attribut xs:type und xs:base sind nur in XML-Schema, Teil 2, definierte Datentypen zulässig; siehe auch kommentierte Definition von tsd:native-Datatype in dieser Dokumentation.
  Logische TSD3 Einschränkungen im Vergleich zu vollständigen DTD-Funktionalitäten, die in einem vollständigen XML-Schema ausgedrückt werden können (z.B. allgemeine Entitäten).
- Enumerationen für Elemente ohne Unterelemente und mit Attributen können nicht definiert werden (Einschränkung für XML-Schema: Dieses kann nicht ohne named complex types ausgedrückt werden.)
  Physische TSD3 Einschränkungen
- Anhängen einer physischen Schema-Definition ist nicht möglich ohne das Element tsd:which unterhalb von rekursiven Elementen.
- Für jedes Schema, Element oder jede attribute-definition ist nur ein schemaInfo-,

elementInfo-, attributeInfo-Element möglich.

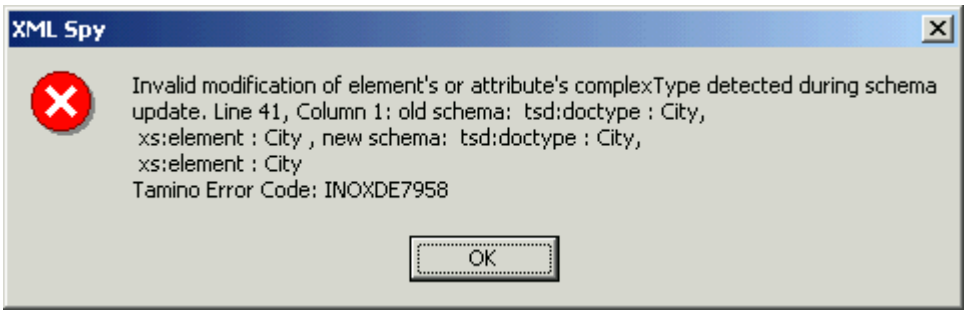
Einschränkungen von XMLSpy und dem Tamino Paket

- Transaktionsorientierte Verarbeitung wird nicht unterstützt.
- Nur XML-Dokumente und XSD-Schema-Dokumente können über XMLSpy in Tamino gespeichert werden.
- Inkludierung von Schemas innerhalb von Schemas wird nicht unterstützt.
- Import von Schemas wird nicht unterstützt.
- Namespaces (Präfixe) in einem Schema werden von Tamino Server nicht unterstützt.
- Die in XMLSpy verfügbaren Tamino-Funktionen sind nicht mit WebDAV kompatibel.

Bearbeiten von Tamino Schemas

Bearbeiten von Tamino Schemas (die XML Dokumente in der Datenbank referenzieren)
Wenn Sie eine Schema-Definition ändern, auf der mehrere XML-Dokumente basieren, kann dies zur Folge haben, dass die verknüpften XML-Dokumente nicht mehr gültig sind. Beim Speichern prüft Tamino, ob die Schemas noch gültig sind. Falls nicht, erhalten Sie eine Tamino-Fehlermeldung.

Beispiel:
Ein Schema und ein verknüpftes Dokument sind in Tamino vorhanden. Das Schema wird in XMLSpy geöffnet, und ein neues obligatorisches Element wird an einen Complex Type angehängt. Wenn Sie dieses Schema nun in Tamino speichern, erhalten Sie folgende Fehlermeldung.



Wenn Sie nun das Element von obligatorisch in optional ändern (min=0 and max=1), kann das Schema in Tamino gespeichert werden, ohne dass die referenzierten XML-Dokumente ungültig werden.

Erstellen von XML-Dateien, die auf Tamino Schemas basieren

So erstellen Sie XML-Dateien, die auf Tamino basieren:
Sie können XML-Dateien, die auf Schemas basieren, auf zwei verschiedene Arten erstellen:

- Verwendung eines bereits existierenden XML-Dokuments als **Vorlage**, für die sowohl die XML-, als auch die Schemadatei bereits in Tamino vorhanden ist, oder
- Verwendung einer Schemadatei, die bereits in Tamino existiert, als Basis für Ihr neues XML-Dokument.

So erstellen Sie eine XML-Datei die auf einer anderen XML (Vorlagen-)Datei basiert:

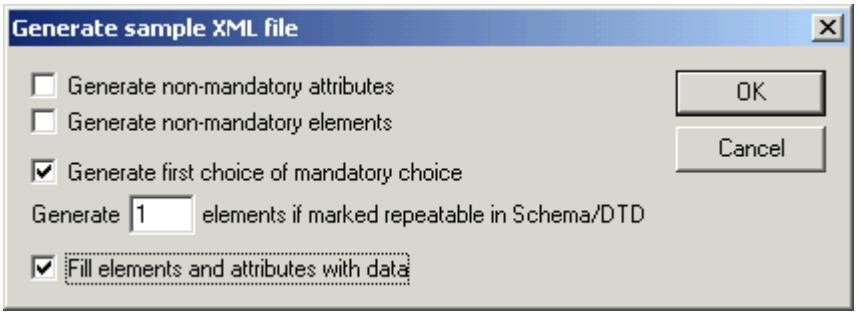
1. Erzeugen Sie eine Abfrage für das XML Dokument, welches als Vorlage verwendet werden soll, und holen Sie das Dokument vom Tamino Server. (Alternativ können Sie auch die Methode Datei holen verwenden.)
2. Bearbeiten Sie das XML-Dokument.
3. Verwenden Sie die Menüoption **DB | Tamino | Dokument hinzufügen**, um das neue XML-Dokument wieder auf dem Tamino Server zu speichern. Beispieldateien finden Sie im Ordner Example, z.B. cars1.xml und cars2.xml.

 © 2007 Altova GmbH

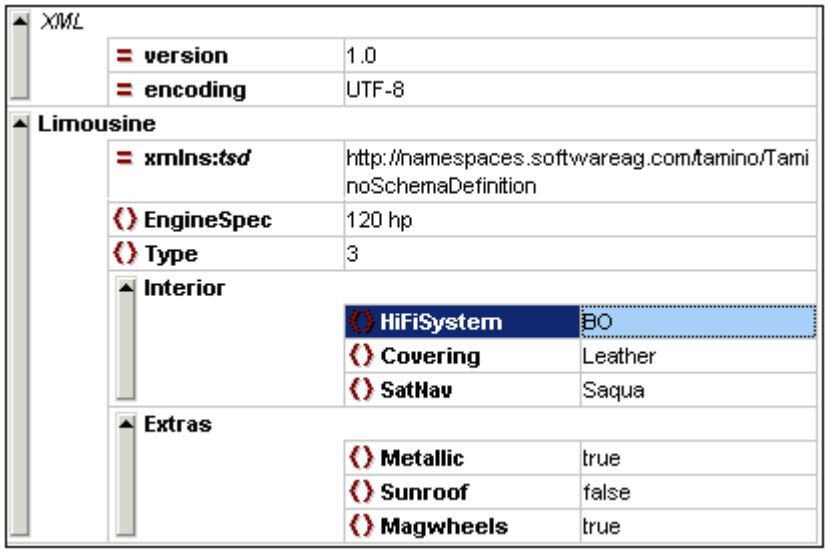
So erstellen Sie eine XML-Datei, die auf einem lokal gespeicherten Tamino Schema basiert

Nachdem Sie die Schemadatei geöffnet haben, die in Tamino gespeichert ist (BMW Schema in diesem Beispiel)

1. Stellen Sie sicher, dass die geöffnete Schemadatei das aktive Dokument ist.
2. Wählen Sie **DB | Tamino | XML Beispieldatei erzeugen**.



3. Ein Dialogfeld wird angezeigt. Aktivieren Sie die Kontrollkästchen "Erste Wahl von obligatorischer Auswahl generieren" und "Elemente und Attribute mit Daten befüllen". Bestätigen Sie mit OK.
4. Eine XML Beispieldatei wird erzeugt. Ändern Sie nun die Daten wie unten angegeben:

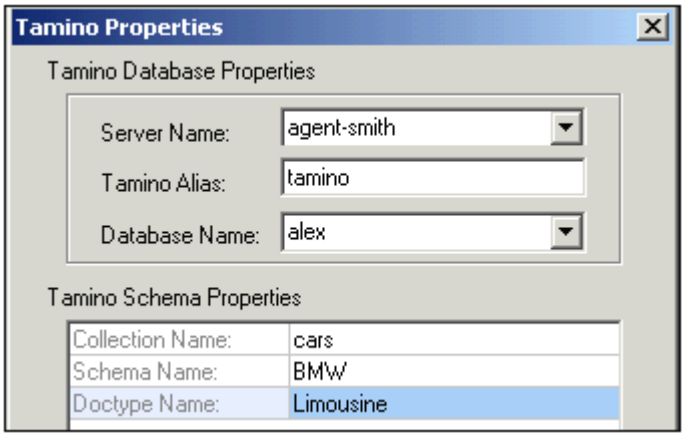


Bitte beachten Sie:

Beim Auswählen der Menüoption **XML | Validieren** wird die XML Datei gegen die **lokale** Tamino Schemadatei validiert (mittels XMLSpy Validator).

So fügen Sie eine XML Datei, einen Datensatz oder ein Dokument in Tamino ein:

1. Wählen Sie die Menüoption **DB | Tamino | Dokument hinzufügen** um das aktive XML-Dokument in Tamino hinzuzufügen.
   Ein Dialogfeld wird geöffnet, in dem Sie um weitere Angaben gebeten werden.
2. Geben Sie die Namen der Datensammlung, des Schemas und des Doctype im Fenster Tamino-Eigenschaften ein, und bestätigen Sie mit OK.



Die XML Datei ist nun mit dem in Tamino gespeicherten Schema verknüpft. Sie erhalten eine Bestätigungsmeldung, dass das XML-Dokument erfolgreich hinzugefügt wurde.

3. Ändern Sie einen der Einträge in der offenen XML-Datei in (metallic=false) und wählen Sie nochmals **DB | Tamino | Dokument hinzufügen**.
   Die bearbeitete XML-Datei wird mit den in Tamino getätigten Einstellungen in die Datenbank aufgenommen. Sie haben nun zwei XML Dateien, die mit der Cars-Datensammlung assoziiert sind, BMW Schema, und Limousine Doctype.

Bitte beachten Sie:

Beim Validieren einer **XML-Datei**, die zum **Tamino Server hinzugefügt** wurde, wird diese gegen die referenzierte Schemadatei in Tamino validiert.

Derzeit können nur XML- und verknüpfte XSD-Dateien über XMLSpy in Tamino gespeichert werden.

Abfrage von Daten in Tamino

Abfrage der Datenbank in Tamino

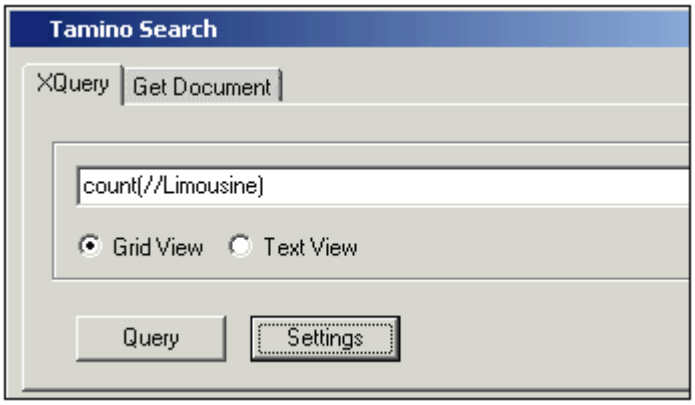
Mit der Menüoption **DB | Tamino | Suche** definieren Sie eine neue Abfrage. Tamino verwendet die Abfragesprache **XQuery**. Die Syntax dieser Sprache folgt den Vorgaben von Standard XPath Notationen.

Sie können über folgendes Dialogfeld sowohl eine XQuery starten, als auch eine spezifische Datei aus der Datenbank (wenn Sie deren ID kennen) abrufen. Das Resultat der Abfrage wird als XML-Dokument in der Grid- oder Text-Ansicht angezeigt.

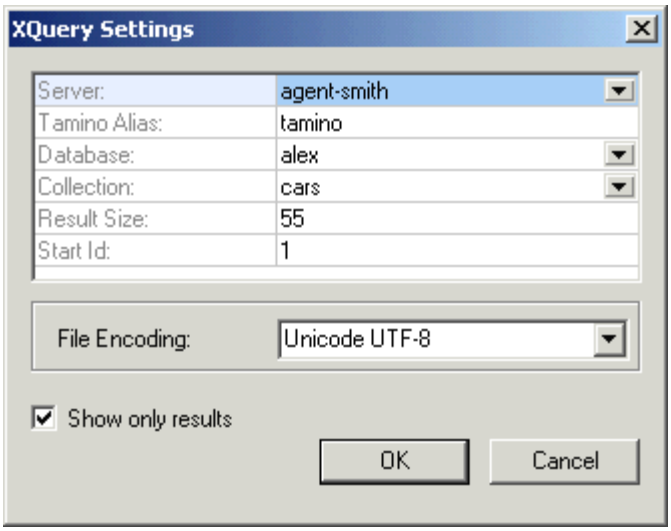
Beispiel:
Sie möchten wissen, wie viele Datensätze es mit dem Doctype "Limousine" gibt. Dafür wird der XPath Befehl **count()** zur Abfrage verwendet.

1. Geben Sie **count(//Limousine)** im XQuery Feld ein.

 © 2007 Altova GmbH

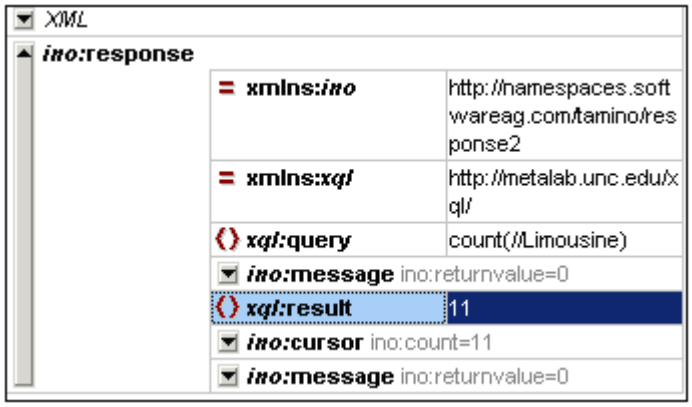


2. Nachdem das Abfragekriterium bestimmt wurde, müssen Sie noch angeben, welche Datenbank, Datensammlung und Doctype durchsucht werden sollen. Klicken Sie auf die Schaltfläche **Einstellungen**, geben Sie die gewünschten Einstellungen an, und bestätigen Sie mit OK. Bitte beachten Sie die **Groß- und Kleinschreibung** bei Ihren Angaben.



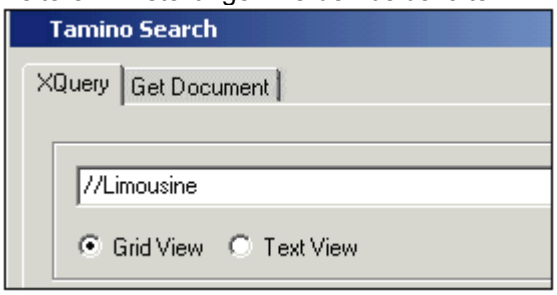
3. Klicken Sie auf die Schaltfläche **Abfrage** und danach **Schließen**, um das Dialogfeld zu schließen.
   Das Resultat der Abfrage wird als XML-Dokument angezeigt: **xql:result** zeigt in diesem Fall 11 an.

   Bitte beachten Sie: Die Standardeinstellung für das Feld Result ist Size: 5. In diesem Beispiel wurde die Zahl auf 55 erhöht. Wenn Sie das Kontrollkästchen "Nur Resultate anzeigen" aktivieren, wird im XML-Dokument nur das Resultat dargestellt.

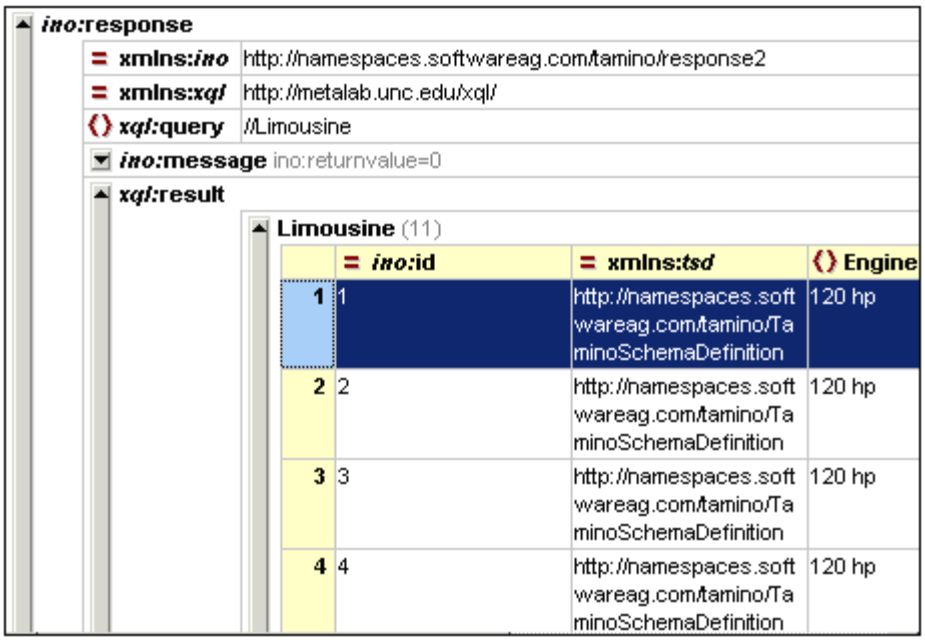


Spezifische Dokumente aus Tamino holen:

1. Definieren Sie eine neue Abfrage, bei der alle XML-Dokumente des Doctypes Limousine angezeigt werden sollen und klicken Sie auf die Schaltfläche Abfrage. Alle weiteren Einstellungen werden beibehalten.

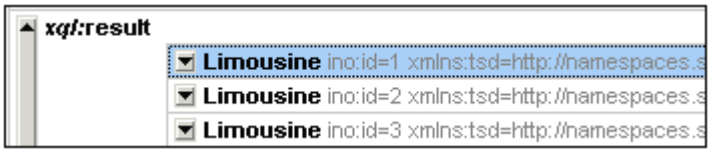


Das Resultat wird angezeigt, in der Tabellen-Ansicht werden mehrere Resultate angezeigt.

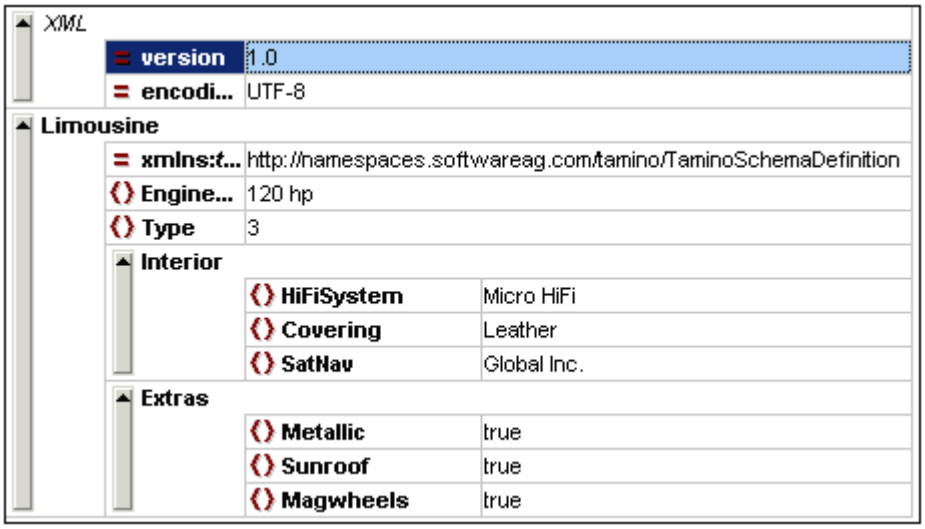


2. Klicken Sie in die Spaltennummer in der Tabelle, und wählen Sie die Menüoption **DB | Tamino | Aktives Dokument holen**. Wenn die Tabellen-Ansicht nicht aktiviert ist, klicken Sie in die entsprechende Zeile "Limousine".

 © 2007 Altova GmbH

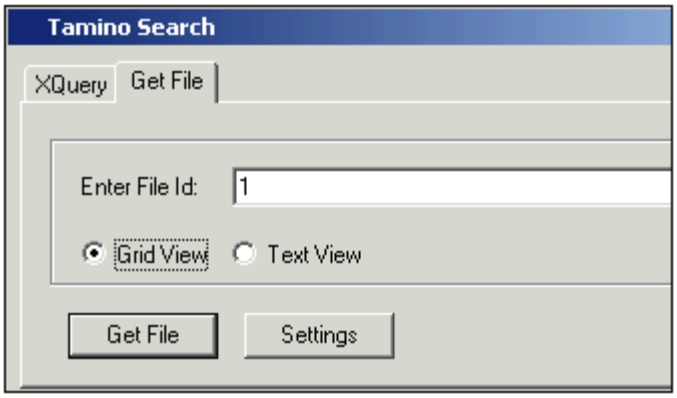


Das Abfrageresultat wird als XML-Dokument im Hauptfenster angezeigt.



Dateimethode zur Suche eines bestimmten Dokuments:

1. Notieren Sie die ID Nr. des Abfragedokuments und öffnen Sie erneut das Dialogfeld für die Abfrage. Die Dokument-ID ist der Eintrag **ino:id=x** .
2. Klicken Sie auf das Register **Dokument holen**, und geben Sie den Tamino Doctype in das Textfeld ein.
3. Geben Sie die Dokumenten-ID in das Textfeld "Dokument-ID eingeben" ein.



4. Klicken Sie auf **Dokument holen** um das XML-Dokument zu öffnen.

So speichern Sie eine Tamino Abfrage:

Verwenden Sie die Tastenkombination **Strg+S** um die Abfrage lokal zu speichern. Hierdurch wird die Abfrage als XML-Dokument gespeichert. Diese steht allerdings nicht mit dem Tamino Server in Verbindung.

Abfrage aktualisieren:

Nachdem Sie eine Abfrage definiert und ausgeführt haben, kann das Abfragedokument verwendet werden, um diese zu einem späteren Zeitpunkt wieder zu verwenden.

1. Gehen Sie zurück zum ursprünglichen Abfragedokument.
2. Wählen Sie die Menüoption **DB | Tamino | Aktualisieren**.
   Tamino wird so mit den gleichen Kriterien neu abgefragt. Das neue Abfrageresultat stellt dann ein aktualisiertes Resultat dar.

Bearbeiten von existierenden XML-Dokumenten und Hinzufügen in Tamino
Ändern Sie Daten im XML-Dokument und wählen Sie die Menüoption **DB | Tamino | Dokument hinzufügen**. Das bearbeitete Dokument wird in Tamino hinzugefügt. Eine Bestätigungsmeldung zeigt an, dass das Dokument erfolgreich hinzugefügt wurde.

15.9.5 Oracle XML DB

Mit XMLSpy können Sie eine Verbindung zu Oracle XML-Datenbanken herstellen, um diese abzufragen.

Folgende Datenbankfunktionen werden unterstützt:

- Hinzufügen (und Registrieren) von XML-Schemas in einer Oracle XML-Datenbank. Zum Registrieren von XML-Schemas mit XMLSpy muss zuvor der Oracle XML DB Client installiert werden.
- Öffnen und Löschen von Schemas
- Abfrage der Datenbank durch Verwendung von XPath Statements (DBUri)
- Durchsuchen von XML-Dokumenten (mittels WebDAV)
- Erstellen von XML-Dokumenten basierend auf Schemas, die in der Datenbank gespeichert sind.

Installation:

- Downloaden und installieren Sie XMLSpy
- Installieren Sie Oracle Server (falls nötig)
- Erzeugen Sie eine Oracle Datenbank

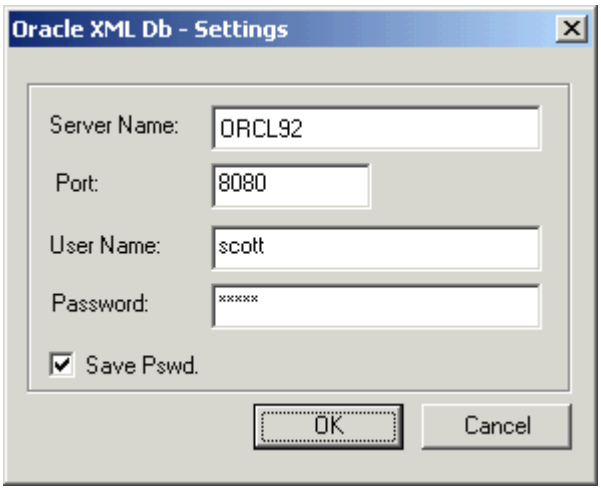
Unter http://otn.oracle.com/sample_code/tech/xml/xmldb/XDBBasicDemo.pdf finden Sie eine Demoinstallation für eine Oracle XML DB. In diesem Dokument wird beschrieben, wie die Oracle Software installiert und für XMLSpy konfiguriert wird.

Suchen...

Mit diesem Befehl können Sie die Oracle XML Datenbank durch Verwendung von DBUri Statements abfragen.

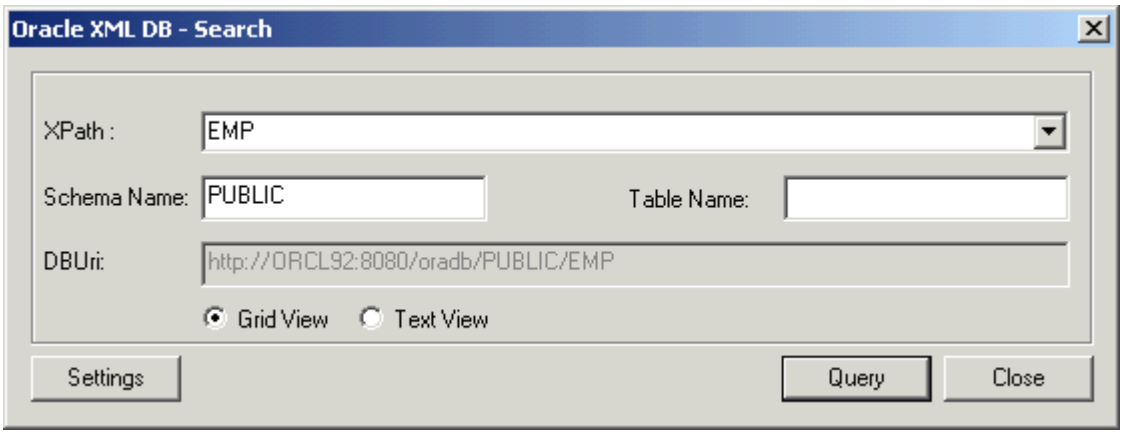
Bei Auswahl dieses Befehls erscheint das folgende Dialogfeld:

© 2007 Altova GmbH



Geben Sie Ihre Oracle-Einstellungen ein und klicken Sie auf **OK**.

Geben Sie die Abfrage als XPath im **XPath** Feld ein. Ihre Abfrage wird im darunter liegenden DBUri Feld angezeigt und wird als http Protokoll an den Server gesendet.



Das Feld **Schema-Name** muss nur bei der ersten Abfrage eingegeben werden. Bei Folgeabfragen wird der Wert automatisch eingetragen.

Im Feld **Tabellenname** können Sie angeben, welche Tabelle abgefragt werden soll.

Einstellungen
Durch Betätigen dieser Schaltfläche wird das Dialogfeld Einstellungen geöffnet, in dem Sie Abfrageeinstellungen tätigen können: Servername, Port, Benutzername und Passwort.

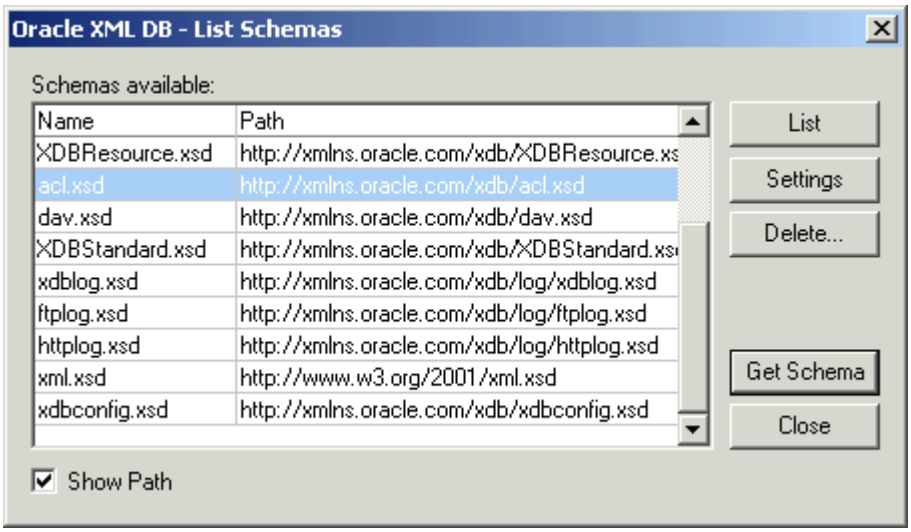
Abfrage
Durch Betätigen dieser Schaltfläche wird die Abfrage abgesendet und das Dialogfeld geschlossen.

Das Resultat der Abfrage erscheint in der Enhanced Grid oder Text-Ansicht, je nachdem, welche Option Sie im Dialogfeld angegeben haben. War die Abfrage nicht erfolgreich, wird eine Fehlermeldung angezeigt.

Schemas auflisten

So zeigen Sie die Liste der verfügbaren Schemas in der Oracle Datenbank an:

1. Wählen Sie die Menüoption **Oracle XML-Datenbank | Schemas auflisten**. Das Dialogfeld Oracle XML-Datenbank - Schemas auflisten wird geöffnet.
2. Klicken Sie auf die Schaltfläche **Auflisten** um die Schemas der aktuellen Datenbank anzuzeigen.



3. Doppelklicken Sie auf eines der Schemas in der Liste oder betätigen Sie die Schaltfläche "Schema holen". Das ausgewählte Schema wird in der Schema/WSDL Design-Ansicht angezeigt.

Das Dialogfeld Oracle XML-Datenbank - Schemas auflisten erlaubt es Ihnen, weitere Optionen auszuwählen.

Einstellungen
Mit dieser Schaltfläche definieren oder ändern Sie die Einstellungen der aktuellen Oracle XML Datenbank.

Löschen
Mit dieser Schaltfläche löschen Sie ein eingetragenes Schema (welches im Listenfeld angezeigt wird) aus der Oracle XML Datenbank.

1. Klicken Sie auf **Löschen.**
2. Wählen Sie entweder das Optionsfeld ADO oder ODBC Connection String und klicken Sie auf die Schaltfläche Erzeugen.
   Das Dialogfeld Datenverknüpfungseigenschaften wird geöffnet (oder wählen Sie das Dialogfeld Datenquelle falls Sie ODBC ausgewählt haben).
2. Wählen Sie **MS OLE DB Provider for Oracle** und klicken Sie auf **Weiter**.
3. Geben Sie die Serverinformationen ein: Servername, Username, Passwort, und klicken Sie auf OK. Der ADO Connection String wird nun im Feld Connection String angezeigt.
4. Klicken Sie auf die Schaltfläche Löschen, um das Schema zu entfernen.

Schema holen
Öffnet das ausgewählte Schema in der Schema/WSDL Design-Ansicht.

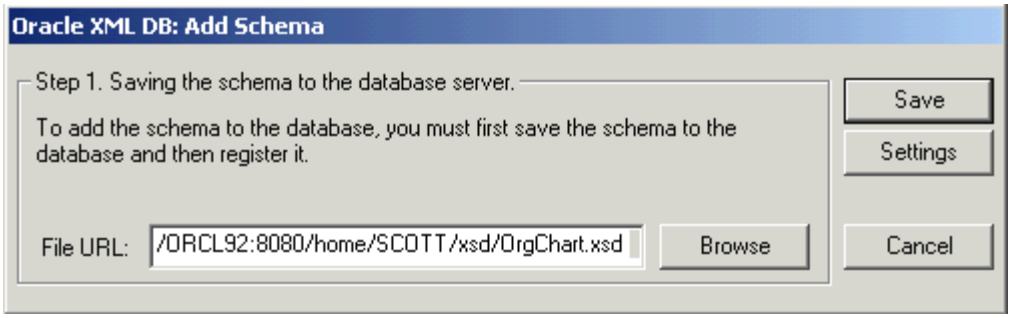
© 2007 Altova GmbH

Schließen
Schließt das Dialogfeld.

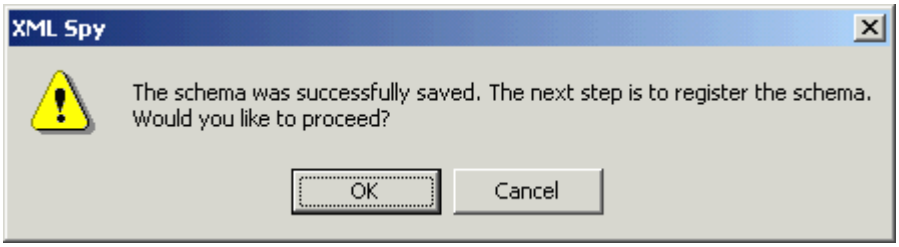
Schema hinzufügen

Mit diesem Befehl können Sie ein XML Schema zur Oracle XML Datenbank hinzufügen. Informationen zum Thema XML-Dokument Instanzdatei in die Datenbank einfügen, finden Sie unter XML-Dokumentinstanzen hinzufügen.

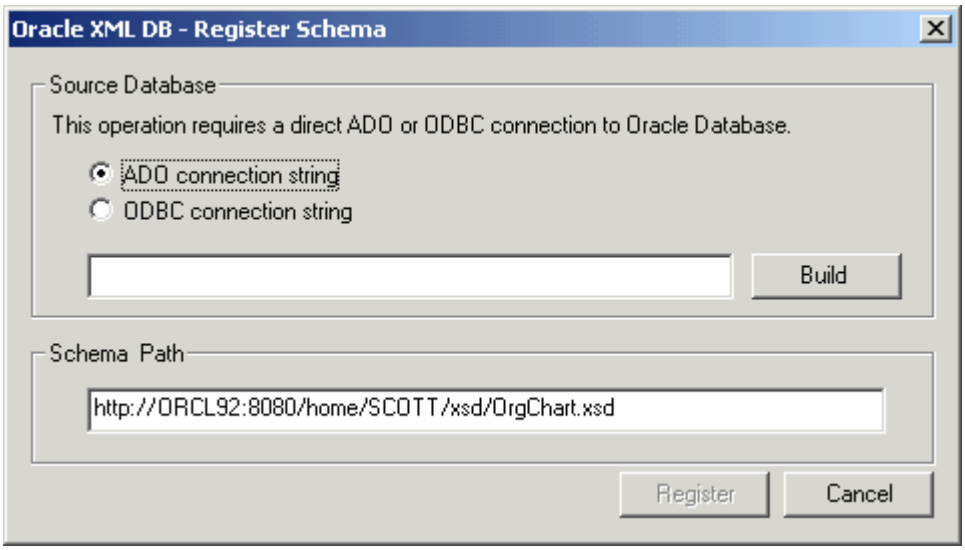
1. Wählen Sie die Menüoption **Oracle XML-Datenbank | Schema hinzufügen**.
   Dies ist der erste Schritt beim Hinzufügen eines Schemas. Das Schema wird hierdurch in der Datenbank gespeichert. Die URL der Datei wird automatisch im Textfeld Datei URL angezeigt.
   Klicken Sie auf die Durchsuchen-Schaltfläche um anzugeben, wo die Datei gespeichert werden soll. Geben Sie den Dateinamen der neuen Schemadatei ein.



2. Klicken Sie auf **Speichern** um das Schema in der Datenbank zu speichern. Danach erhalten Sie eine Meldung, dass das Schema erfolgreich in der Datenbank gespeichert wurde. Mit OK registrieren Sie nun das Schema.



   Das Dialogfeld **Schema registrieren** wird geöffnet. Der Pfad des soeben gespeicherten Schemas wird im Textfeld Schemapfad angezeigt.
3. Klicken Sie auf das Optionsfeld ADO oder ODBC Connection String und dann die Schaltfläche Erzeugen.



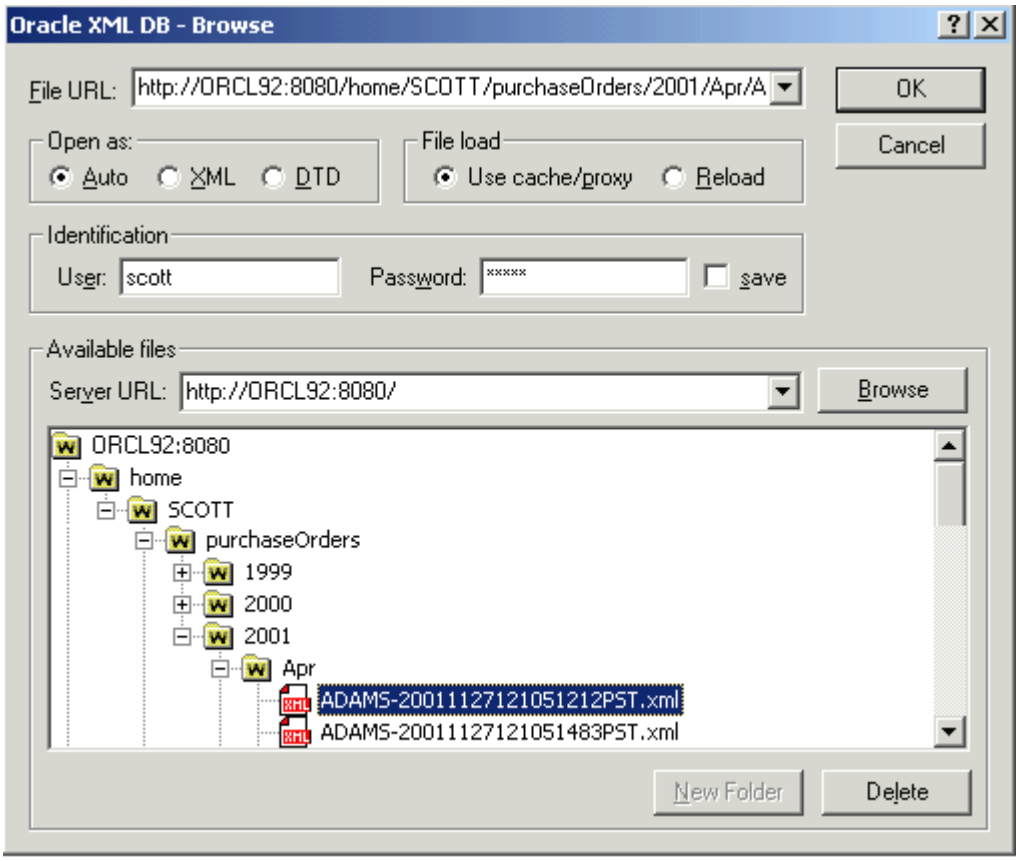
Hierdurch wird das Dialogfeld Datenverknüpfungseigenschaften geöffnet (oder wählen Sie das Dialogfeld Datenquelle, falls Sie ODBC ausgewählt haben).

4. Wählen Sie **MS OLE DB Provider für Oracle** und klicken Sie auf **Weiter**.
5. Geben Sie die Serverinformationen ein: Server Name, Benutzername und Passwort, klicken Sie auf das Kontrollkästchen **Speichern des Kennworts zulassen**, und klicken Sie auf OK.
   Der ADO Connection String wird nun im Textfeld Connection String angezeigt.
6. Klicken Sie auf **Registrieren**.
   Jetzt ist das Schema in der Oracle XML DB registriert.

Oracle XML-Dokumente durchsuchen

Mit diesem Befehl können Sie die XML-Dokumente durchsuchen, die auf Ihrem Server zur Verfügung stehen. Serverinformationen, die von früheren Datenbank-Abfragen zur Verfügung stehen, werden automatisch eingegeben. Sollte dies nicht der Fall sein, geben Sie die Informationen händisch ein.

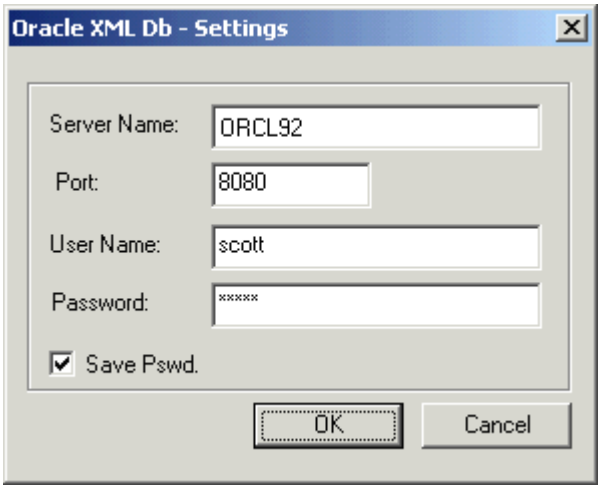
© 2007 Altova GmbH



Im Baumdiagramm finden Sie die spezifischen XML Dateien. Doppelklicken Sie auf das Baumdiagramm um es weiter auszuklappen. Sie können auch auf eine Datei doppelklicken und die Schaltfläche OK betätigen. Mit **Neuer Ordner** können Sie neue Ordner einfügen, mit **Löschen** entfernen Sie zuvor ausgewählte XML Dateien.

Eigenschaften

Mit diesem Befehl zeigen Sie die Eigenschaften der Oracle XML DB für die aktuelle Abfrage, das Schema oder XML-Dokument im Hauptfenster an.



Server-Name
Name des Servers mit dem Sie über http Protokoll verbunden sind.

Port
Serverport, der verwendet wird, um die Serververbindung herzustellen. 8080 ist die localhost-Verbindung.

Benutzername
Ihr Username.

Passwort
Ihr Passwort.

Passw.speichern
Speichert das Passwort für dieses XML-Dokument.

XML Dokumentinstanzen hinzufügen

Hinzufügen einer XML-Dokumentinstanz in die Datenbank:

1. Öffnen Sie ein bereits vorhandenes Schema aus der Oracle XML DB durch Verwendung des Befehls **DB | Oracle XML DB |Schema auflisten**.
2. Wählen Sie die Menüoption **DTD/Schema | XML Beispieldatei erzeugen**.
3. Wählen Sie die spezifischen Optionen für die XML Datei und klicken Sie auf **OK**. Hierdurch wird die XML Datei erstellt, die mit dem Schema assoziiert ist (bitte lesen Sie die Anmerkung am Ende dieses Kapitels).
4. Wählen Sie die Menüoption **Datei | Speichern unter URL**.
5. Klicken Sie auf die Schaltfläche **Durchsuchen** und navigieren Sie zu dem Ordner, in dem Sie das XML-Instanzdokument speichern möchten. Mit Hilfe der Schaltfläche **Neuer Ordner** können Sie einen neuen Ordner anlegen.
6. Geben Sie den Namen der Datei nach dem Pfad/URL im Feld Datei URL an (bitte stellen Sie sicher, dass die Datei auf dem Oracle Datenbank Server gespeichert wird).
7. Klicken Sie auf die Schaltfläche OK um das XML Instanzdokument an dieser Stelle zu speichern.

Bitte beachten Sie:
Da der Root-Node jedes Dokuments ein **noNamespaceSchemalocation** Attribut enthält, welches es als Instanz eines registrierten Schemas identifiziert, werden die Dokumente auseinander genommen und als Objekte in der Datenbank gespeichert.

15.9.6 Datenbank abfragen

Enter topic text here.

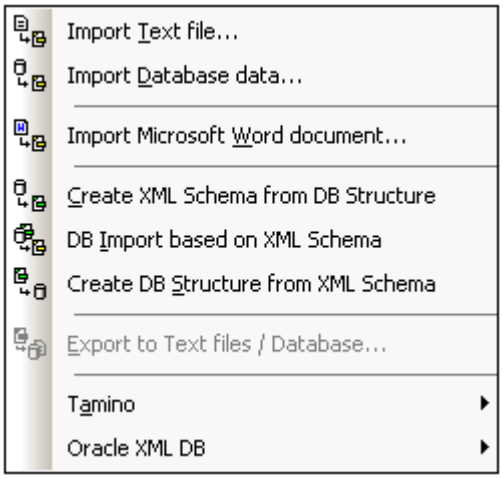
© 2007 Altova GmbH

15.10 Menü "Konvertieren"

Mit XMLSpy stehen Ihnen leistungsfähige Datenaustauschfunktionen zur Verfügung:

- Importieren und Exportieren von Text-, Word-Prozessor-, Datenbanken- und XML-Dateien.
- Erzeugen von XML-Schemas, die auf einer bereits existierenden Datenbank **basieren**.
- Importieren von Datenbankdaten, die auf einem existierenden XML Schema basieren.
- Erzeugen einer **Datenbankstruktur**, die auf einem existierenden Schema basiert.

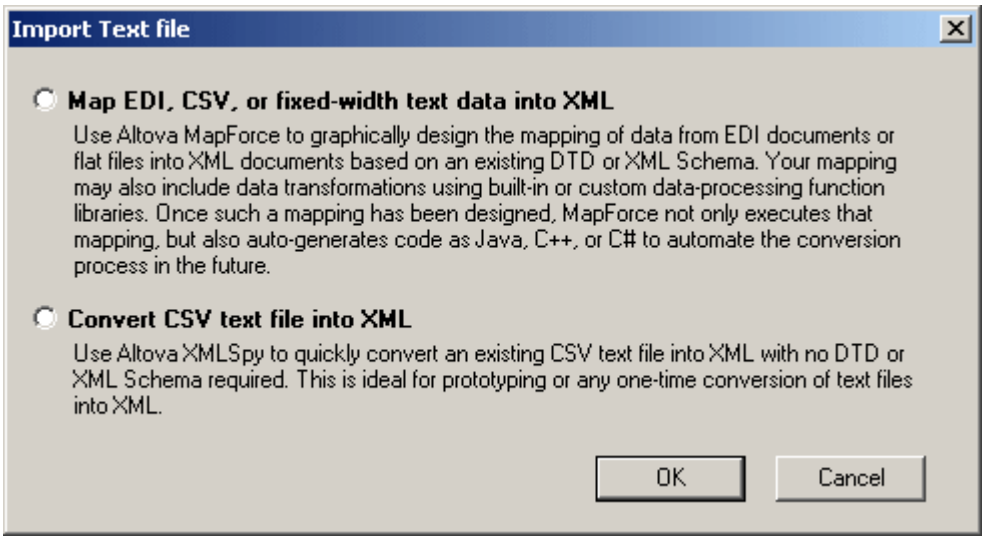
Diese Funktionen stehen im Menü **Konvertieren** zur Verfügung (*siehe Screenshot*).



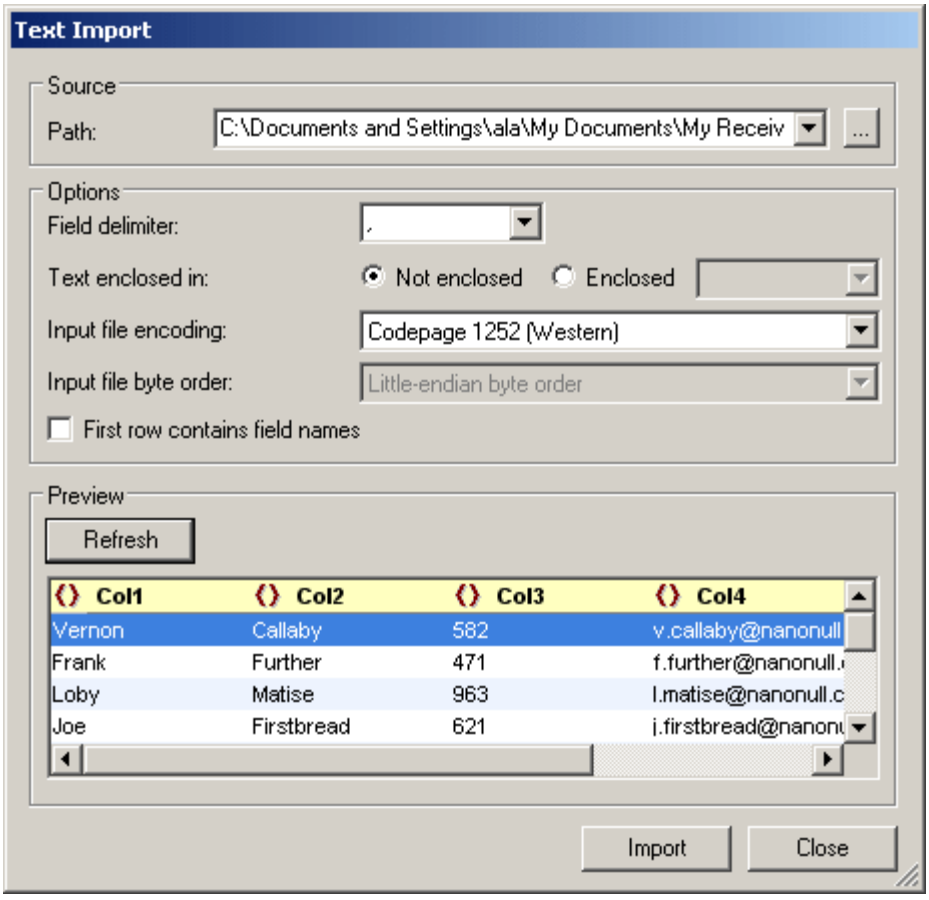
15.10.1 Textdatei importieren

Mit diesem Befehl lässt sich jede Datei mit **strukturierten Text** in XMLSpy importieren und sofort ins XML-Format konvertieren. Diese Funktion ist nützlich, um Altdatenbestände aus älteren Systemen zu importieren. Die Schritte zum Importieren von Daten in einer Textdatei als XML-Dokument sind unten beschrieben.

1. Wählen Sie den Menübefehl **Konvertieren | Textdatei importieren**. Das folgende Dialogfeld wird angezeigt:



2. Wählen Sie eine der folgenden Optionen:
   - EDI-, CSV- oder Textdaten mit fester Länge auf XML mappen (Sie müssen Altova MapForce installiert haben, um diese Option auszuwählen)
   - CSV-Textdatei in XML konvertieren
3. Klicken Sie auf **OK**. Daraufhin wird das Dialogfeld "Textimport" (*Abbildung unten*) angezeigt. Die Verwendung dieses Dialogfelds wird unten beschrieben.



© 2007 Altova GmbH

Pfad
Geben Sie den Pfad zu der zu importierenden Datei in das Textfeld "Pfad" ein oder wählen Sie die Datei mit Hilfe der Durchsuchen-Schaltfläche rechts vom Textfeld aus. Nachdem Sie die Datei ausgewählt haben, wird eine Grid-Vorschau der XML-Datei im Fenster "Vorschau" angezeigt. Alle Änderungen an den in diesem Dialogfeld ausgewählten Optionen, werden sofort in der Vorschau angezeigt.

Feld-Trennzeichen
Um eine Textdatei erfolgreich importieren zu könnenn, müssen Sie das Feld-Trennzeichen angeben, das zum Trennen von Spalten oder Felder in der Datei verwendet wird. XMLSpy erkennt gebräuchliche Zeilen-Trennzeichen automatisch (CR, LF oder CR+LF).

Text eingeschlossen in
In Textdateien, die aus alten Datenbeständen exportiert wurden, sind Textwerte manchmal zur besseren Unterscheidung von numerischen Werten in Anführungszeichen eingeschlossen. In diesem Fall können Sie angeben, welche Art von Anführungszeichen in Ihrer Datei verwendet werden, damit diese Anführungszeichen beim Import der Daten automatisch entfernt werden.

Eingabedatei-Encoding
Die Daten werden ins Unicode Format (auf dem alle XML-Dokumente basieren) konvertiert, daher müssen Sie angeben, in welchem Zeichensatz die Datei derzeit kodiert ist. Bei amerikanischen oder westeuropäischen Windows-Systemen ist dies meist Codepage 1252, auch als ANSI Encoding bezeichnet.

Eingabedatei-Bytefolge
Bei Import von 16-Bit oder 32-Bit Unicode (UCS-2, UTF-16 oder UCS-4) Dateien, können Sie auch zwischen der Bytefolge Little Endian und Big Endian wählen.

Erste Zeile enthält Feldnamen
Es kommt außerdem bei Textdateien häufig vor, dass diese in der ersten Zeile der Datei die Feldnamen enthalten. Aktivieren Sie dieses Kontrollkästchen, wenn das der Fall ist.

Vorschau
Im Bereich "Vorschau" können Sie die Spaltenüberschriften umbenennen, indem Sie auf den Namen klicken und ihn bearbeiten. Die Spaltenüberschriften werden im XML-Dokument die Element- oder Attributnamen sein. Sie können auswählen, ob eine Spalte im XML-Dokument ein Element oder ein Attribut werden soll oder ob sie nicht in das XML-Dokument importiert werden soll. Im obigen Screenshot ist `Col1` ein Element, `Col2` ein Attribut und `Col3` wird nicht importiert.

Wenn Sie mit den Optionen zufrieden sind, klicken Sie auf **Import**. Die importierten Daten werden in ein XML-Dokument konvertiert, das in der Grid-Ansicht angezeigt wird.

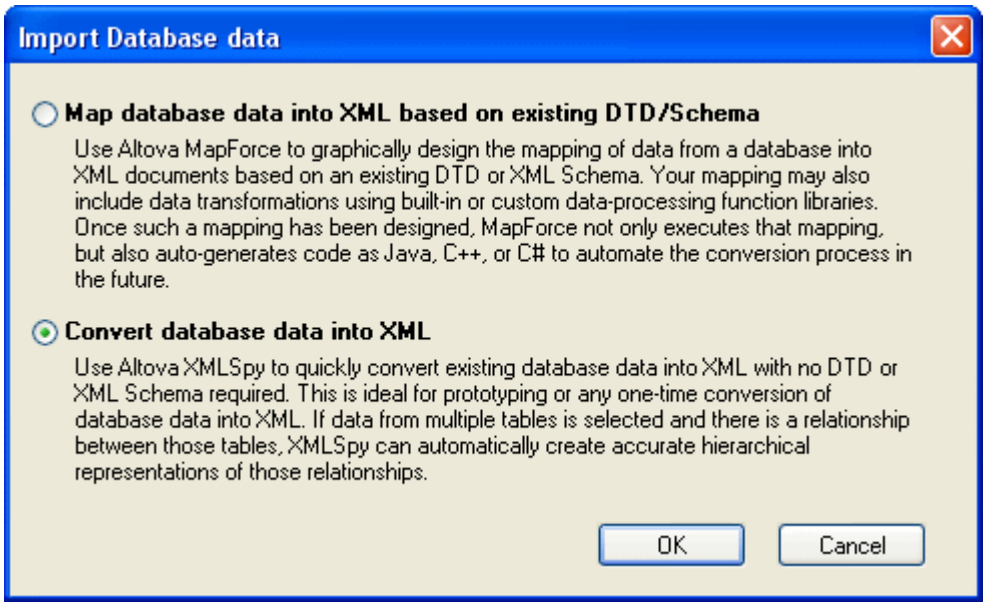
15.10.2 Datenbankdaten importieren

Mit Hilfe des Befehls **Datenbankdaten importieren** können Sie Daten aus verschiedenen Datenbanktypen in eine XML-Datei importieren. Der Import erfolgt in zwei Schritten:

1. Aufbau einer Verbindung zur Datenbank
2. Auswahl der zu importierenden Daten

Um Datenbankdaten zu importieren, gehen Sie folgendermaßen vor:

1. Wenn Sie auf den Befehl "Datenbankdaten importieren" klicken, wird das gleichnamige Dialogfeld (*Abbildung unten*) geöffnet.

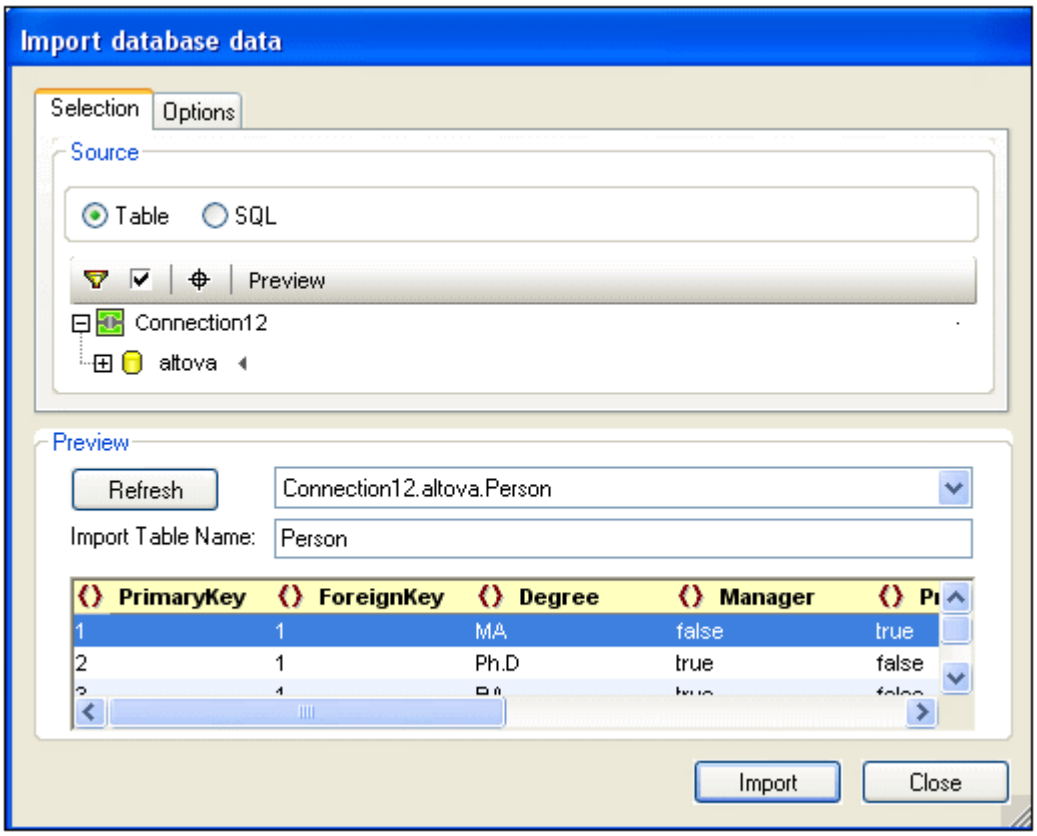


2. Wählen Sie die Option "Datenbankdaten in XML konvertieren" aus und klicken Sie auf **OK**. Daraufhin wird das Dialogfeld "Connect to a Data Source" geöffnet. (Wenn Sie die Option "Datenbankdaten auf Basis eine/r/s DTD/Schemas auf XML mappen" auswählen, muss Altova MapForce installiert sein, damit das Mapping durchgeführt werden kann.)
3. Stellen Sie im Dialogfeld Connect to Data Source eine Verbindung zur Datenbank her. Eine Anleitung zum Herstellen einer Verbindung zu unterschiedlichen Datenbanktypen finden Sie im Abschnitt Herstellen einer Verbindung zu einer Datenquelle.
4. Nachdem Sie die Verbindung hergestellt haben, werden im Dialogfeld "Datenbankdaten importieren" Register und Fenster angezeigt, über die Sie die gewünschten Datenbankdaten auswählen können. Diese Optionen sind weiter unten beschrieben. Nachdem Sie damit fertig sind, klicken Sie auf die Schaltfläche **Import,** um die Datenbankdaten zu importieren.

© 2007 Altova GmbH

Datenauswahl und Importoptionen
Das Dialogfeld "Datenbankdaten importieren" zum Auswählen der Daten und Einstellen der Importoptionen besteht aus zwei Teilen (*die in den Abbildungen unten separat gezeigt werden*):

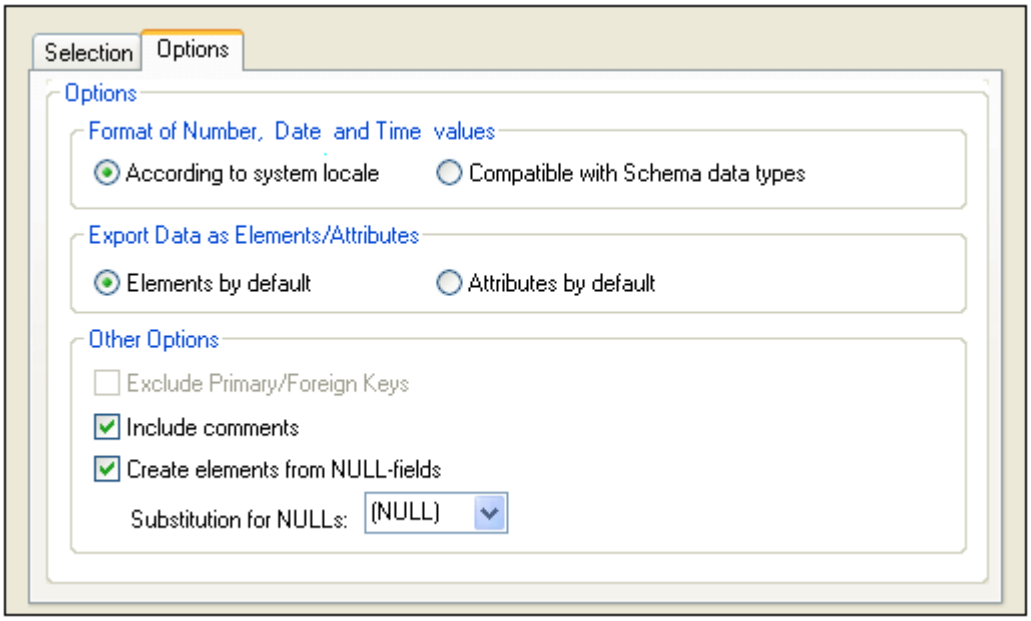
- einem oberen Bereich mit zwei Registern: (i) Auswahl und (ii) Optionen.
- einem unteren Bereich, dem Vorschaufenster, in dem die Daten entsprechend der getroffenen Auswahl und der gewählten Optionen angezeigt werden.



Register "Auswahl"
Auf dem Register "Auswahl" (*Abbildung oben*) werden entweder die Tabellen der Datenbank angezeigt oder Sie sehen eine editierbare SQL-Anweisung zur Auswahl der gewünschten Tabellen.

In der Tabellenansicht können Sie die Tabellen in der Datenbank auswählen, die Sie importieren möchten. Aktivieren Sie dazu das Kästchen neben der jeweiligen Tabelle (*Abbildung oben*). Der Inhalt der Tabelle wird im Vorschaufenster angezeigt. Die Tabellenauswahl kann im Vorschaubereich noch weiter gefiltert werden (*siehe unten*).

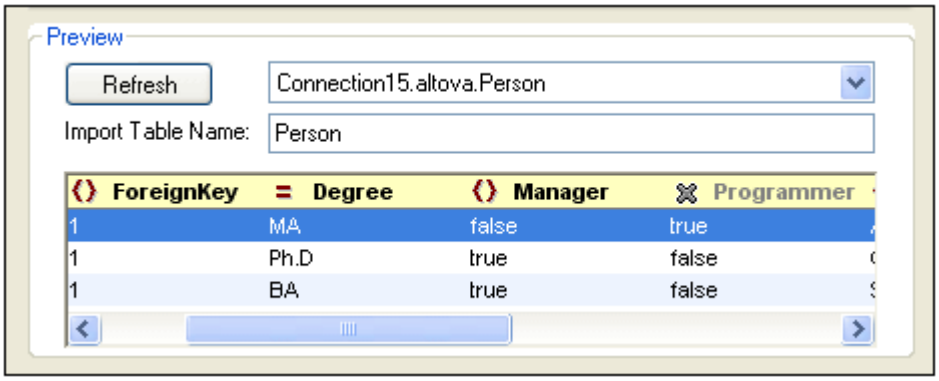
Register "Optionen"
Auf dem Register "Optionen" (*Abbildung unten*) können Sie definieren, wie Zahlen, Datums- und Uhrzeitwerte importiert werden sollen, ob Daten als Elemente oder Attribute importiert werden sollen und ob Kommentare und `NULL` Felder mitimportiert werden sollen.



Wenn `NULL`-Felder für den Import aktiviert sind, können Sie dafür einen XML-Ersetzungswert eingeben.

Vorschaubereich
Im Vorschaubereich (*Abbildung unten*) wird die Struktur der auf dem Register "Auswahl" ausgewählten Tabelle angezeigt. Wenn auf dem Register "Auswahl" eine neue Tabelle ausgewählt wird, klicken Sie auf die Schaltfläche "Aktualisieren", um die Vorschau zu

aktualisieren.



Sie können festlegen, ob ein Feld als Element oder Attribut oder gar nicht importiert werden soll. Klicken Sie dazu auf das Symbol links vom Spaltennamen. Klicken Sie darauf, um der Reihe nach die Optionen "Element", "Attribut" und "Ignorieren" zu aktivieren. In der Abbildung oben wurde das Feld `Manager` für den Import als Element und das Feld `Degree` als Attribut definiert, während das Feld `Programmer` nicht importiert wird.

Datentyp-Konvertierungen
Informationen zu Konvertierung von Datenbank-Datentypen in XML-Schema-Datentypen finden Sie in den Anhängen.

15.10.3 MS Word Dokument Importieren

Mit diesem Befehl können Sie jedes beliebige **Word Dokument importieren und in ein** XML-Format konvertieren, wenn Sie Absatzformatvorlagen in Microsoft Word verwenden. Für diese Option wird Microsoft Word oder Microsoft Office (Version 97 oder 2000) benötigt.

Mit diesem Befehl öffnen Sie das Dialogfeld "Öffnen". Wählen Sie das zu importierende Word Dokument aus.

XMLSpy erzeugt automatisch ein XML-Dokument, das CSS Stylesheet enthält. Jeder Absatz in Word generiert ein XML-Element, dessen Name der Name des entsprechenden Absatzformats in Microsoft Word ist.

15.10.4 XML-Schema von DB-Struktur erzeugen

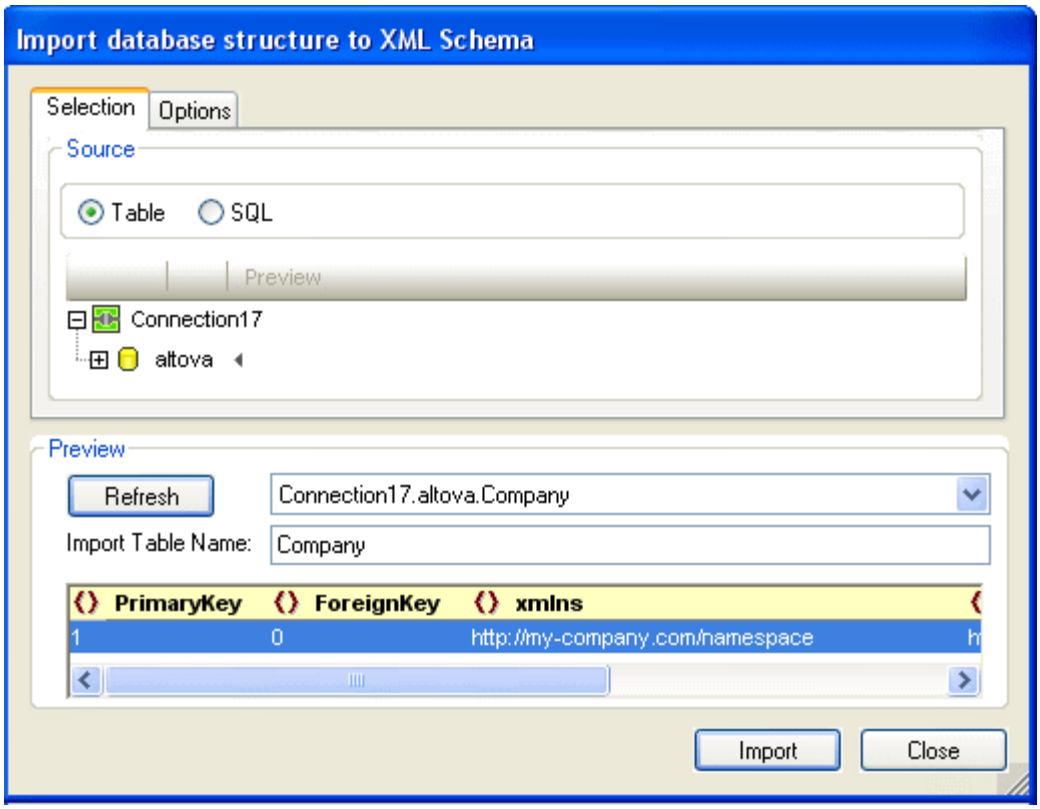
Mit Hilfe des Befehls **XML-Schema von DB-Struktur erzeugen** können Sie anhand der Struktur einer Datenbank ein XML-Schema erzeugen. Es werden die verschiedensten Datenbanktypen unterstützt. Die Erstellung des XML-Schemas erfolgt in zwei Schritten:

1. Aufbau einer Verbindung zur Datenbank
2. Auswahl der Optionen für die Datenbankdaten und Angabe des XML-Schemas

Um ein XML-Schema anhand einer DB-Struktur zu erzeugen, gehen Sie folgendermaßen vor:

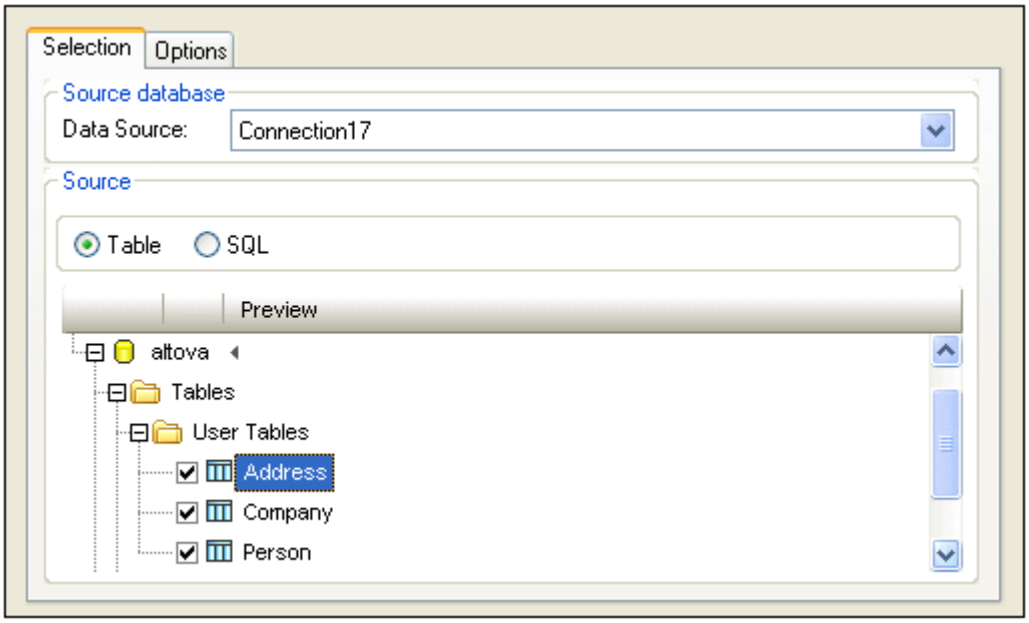
1. Wenn Sie auf den Befehl **XML-Schema von DB-Struktur erzeugen** klicken, wird das Dialogfeld Verbindung zur Datenquelle herstellen aufgerufen.

2. Stellen Sie in diesem Dialogfeld eine Verbindung zur Datenbank her. Eine Anleitung zum Herstellen einer Verbindung zu verschiedenen Datenbanktypen finden Sie im Abschnitt Herstellen einer Verbindung zu einer Datenquelle.
3. Nachdem Sie die Verbindung hergestellt haben, wird das Dialogfeld "Datenbankstruktur in XML-Schema importieren" (*Abbildung unten*) geöffnet, in dem Register und Fenster angezeigt werden, über die Sie die gewünschte Datenbankstruktur auswählen können. Diese Optionen sind weiter unten beschrieben. Klicken Sie anschließend auf die Schaltfläche **Import,** um die Datenbankdaten zu importieren.



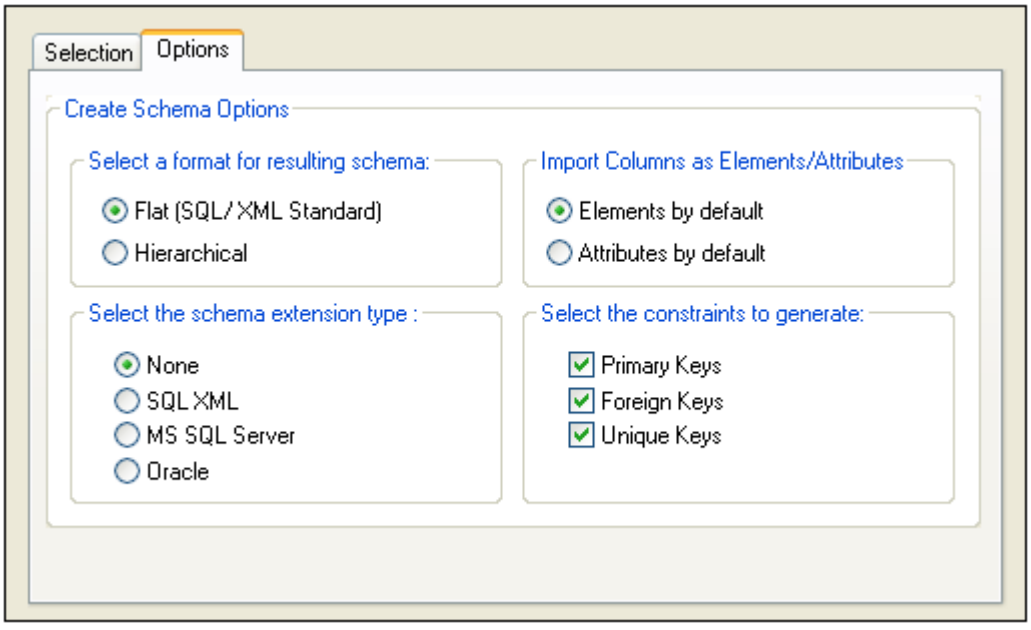
Register "Auswahl"
Auf dem Register "Auswahl" (*Abbildung oben*) wird die Datenquelle im Bereich "Datenbankquelle" aufgelistet. Im Bereich "Quelle" (*Abbildung unten*) werden entweder die Tabellen der Datenbank angezeigt oder Sie sehen eine editierbare SQL-Anweisung zur Auswahl der gewünschten Tabellen.

© 2007 Altova GmbH



In der Tabellenansicht können Sie die Tabellen in der Datenbank auswählen, die Sie importieren möchten. Aktivieren Sie dazu das Kästchen neben der jeweiligen Tabelle (*Abbildung oben*). Der Inhalt der Tabelle wird im Vorschaufenster angezeigt. Die Tabellenauswahl kann im Vorschaubereich noch weiter gefiltert werden (*siehe unten*).

Register "Optionen"
Auf dem Register "Optionen" (*Abbildung unten*) können Sie das Schemaformat, seinen Extension-Typ und die im Schema zu generierenden Datenbank-Constraints definieren und festlegen, ob Spalten als Elemente oder Attribute importiert werden sollen.



Schema-Format: Sie können zwischen einem flachen Format (SQL/XML-Standard) und einem hierarchischen Schemaformat auswählen

- Das **flache** Schema-Modell basiert auf einem ISO-ANSI Working Draft mit dem Titel "XML-Related Specification (SQL/XML)" (im Weiteren als "SQL/XML Working Draft" bezeichnet). Darin wird definiert, wie Datenbanken auf XML gemappt werden. Beziehungen werden im Schema mittels Identity Constraints definiert und es gibt keine Referenzen auf Elemente. Daher hat das Schema eine flache Struktur, die einer Baumstrukturansicht der Datenbank ähnelt. Nähere Informationen zum SQL/XML Working Draft finden Sie unter: http://www.sqlx.org/SQLXdocs
- Im **hierarchischen** Schema-Modell werden die Tabellenabhängigkeiten visuell in Form einer Baumstrukturansicht dargestellt, in der abhängige Tabellen im Content Model als eingerückte Child-Elemente angezeigt werden. Die Tabellenabhängigkeiten werden auch auf dem Register "Identity Constraints" angezeigt.
  Die Tabellen werden im Schema als globale Elemente aufgelistet und die Spalten sind die Elemente oder Attribute dieser globalen Elemente. (Der Benutzer entscheidet, ob die Spalten als Elemente oder Attribute gemappt werden sollen). Beziehungen werden auf hierarchische Art erstellt, sodass ein Sekundärschlüssel in einer Tabelle eine Referenz zu dem globalen Element ist, das für die Tabelle steht.

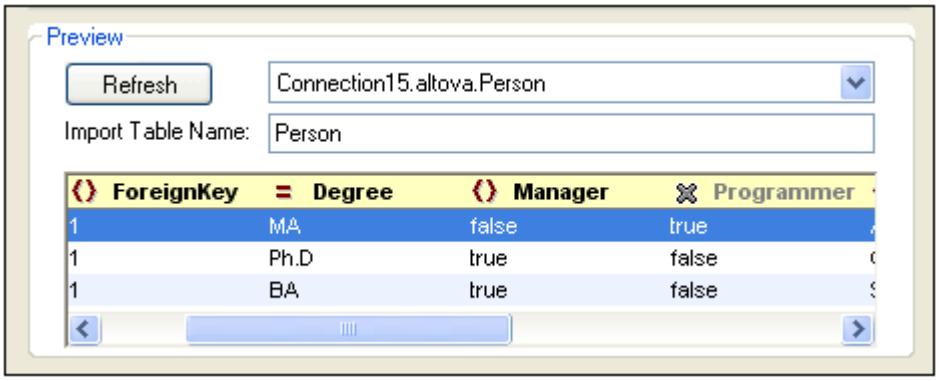
Schema-Extension-Typ: Schema-Extension-Informationen sind zusätzliche Informationen, die aus einer Datenbank ausgelesen werden und die anschließend in Form von Annotationsdaten oder Attributen in das Schema eingebettet werden. Beim Generieren von Schemas gibt es vier Extension-Typ-Optionen: (i) Keine Extension-Informationen; (ii) SQL/XML Extensions; (iii) MS SQL Server Extensions; and (iv) Oracle Extensions. Die folgenden Optionen stehen zur Verfügung:

- **Kein**: Aus der Datenbank werden keine zusätzlichen Infomationen bereitgestellt.
- **SQL XML:** SQL/XML Extensions werden nur eingefügt, wenn Schemas im flachen Format generiert werden. Die Extension-Informationen werden in Form von Annotationen gespeichert und im SQL/XML Working Draft beschrieben.
- **MS SQL Server:** Bei Auswahl von Microsoft SQL Server werden SQL Server Extensions generiert. Nähere Informationen dazu finden Sie online in den SQL Server Books unter *Using Annotations in XSL Schemas* (Unter http://www.microsoft.com/sql/techinfo/productdoc/2000/books.asp). Im Schema wird die folgende Untergruppe mit Annotationsdaten generiert: `sql:relation`, `sql:field`, `sql:datatype`, `sql:mapped`.
- **Oracle:** Beim Arbeiten mit Oracle-Datenbanken werden standardmäßig Oracle-Extensions ausgewählt. Zusätzliche Datenbankinformationen werden als Attribute gespeichert. Eine ausführliche Beschreibung dieser Attribute finden Sie unter http://download-west.oracle.com/docs/cd/B10501_01/appdev.920/a96620/xdb05obj.htm#1027227 (Um diese Seite aufrufen zu können, benötigen Sie Zugriff auf das Oracle Technology-Netzwerk.) Die folgende Attribut-Untergruppe wird derzeit generiert: `SQLName`, `SQLType`, `SQLSchema`.

Anmerkung: SQL Server und Oracle Extensions können zwar für die jeweilige Datenbank generiert werden, unterliegen aber keinen Einschränkungen. Dies erweist sich v.a. beim Arbeiten mit einer dritten Datenbank als nützlich, wenn Sie ein Schema generieren möchten, das später mit SQL Server oder Oracle verwendet werden soll.

Vorschaubereich
Im Vorschaubereich (*Abbildung unten*) wird die Struktur der derzeit auf dem Register "Auswahl" ausgewählten Tabelle angezeigt. Wenn auf dem Register "Auswahl" eine neue Tabelle ausgewählt wird, klicken Sie im Vorschaubereich auf die Schaltfläche "Aktualisieren", um die Vorschau zu aktualisieren.

 © 2007 Altova GmbH



Sie können festlegen, ob ein Feld als Element oder Attribut oder gar nicht importiert werden soll. Klicken Sie dazu auf das Symbol links vom Spaltennamen. Klicken Sie darauf, um der Reihe nach die Optionen "Element", "Attribut" und "Ignorieren" zu aktivieren. In der Abbildung oben wurde das Feld `Manager` für die Generierung als Element und das Feld `Degree` als Attribut definiert, während das Feld `Programmer` nicht generiert wird.

Datentyp-Konvertierungen
Informationen zu Konvertierung von Datenbank-Datentypen in XML-Schema-Datentypen finden Sie in den Anhängen.

15.10.5 DB-Import auf Basis von XML-Schema

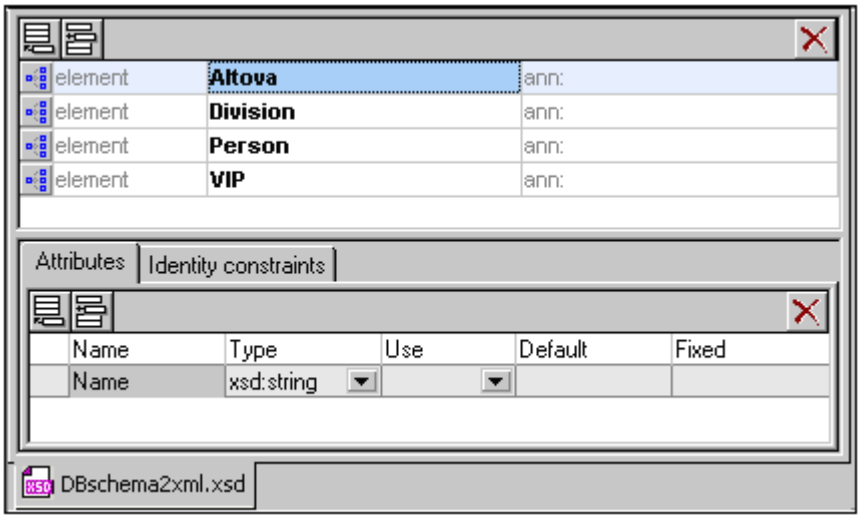
Mit Hilfe des Befehls **DB-Import auf Basis von XML-Schema** wird ein XML-Dokument erstellt, welches gemäß einem bestimmten XML-Schema gültig ist und aus einer Datenbank importierte Daten enthält. Diese Funktion unterstützt die folgenden Datenbanken:

- Microsoft Access 2000 und 2003
- Microsoft SQL Server
- Oracle
- MySQL
- Sybase
- IBM DB2

Welche Daten importiert werden, hängt davon ab, welche Tabelle in der Datenbank ausgewählt wurde. Stellen Sie eine Verbindung zur Datenbank her, wobei das XML-Schema, auf Basis dessen der Import durchgeführt werden soll, in der Schema-Ansicht aktiv sein muss. Wählen Sie anschließend die gewünschte(n) Tabelle(n) aus und klicken Sie auf "Import". Die Daten werden in ein XML-Dokument importiert. Das Dokument hat die Struktur der Schemas, das beim Import der Daten aktiv war.

Im unten gezeigten Beispiel werden Daten aus der MS Access Datenbank `DB2schema.mdb` importiert, während in der Schema-Ansicht das XML-Schema `DBschema2xml.xsd` aktiv ist. Beide Dateien befinden sich im Ordner `Examples/Tutorial` Ihres Applikationsordners. Im folgenden sind die Schritte beschrieben, die beim Import durchgeführt werden müssen:

1. Öffnen Sie die Schema-Datei `DBschema2xml.xsd` in der Schema-Ansicht (*Abbildung unten*).



2. Wählen Sie den Menübefehl **DB-Import auf Basis von XML-Schema**. Daraufhin wird das Dialogfeld Verbindung zur Datenquelle herstellen geöffnet.
3. Wählen Sie die Option Microsoft Access (ADO) und klicken Sie auf **Weiter**.
4. Klicken Sie auf **Durchsuchen** und wählen Sie die Datenbankdatei `DB2schema.mdb` und klicken Sie anschließend auf **Weiter**.
5. Klicken Sie im Dialogfeld "DB-Import auf Basis von XML-Schema" auf das Registrer "Tabellen", wählen Sie eine oder mehrere Tabellen aus, die Sie importieren möchten (z.B. `Altova`) und klicken Sie anschließend auf **Import**. Die Tabelle wird in ein XML-Dokument importiert, das in der Grid-Ansicht angezeigt wird.

Datentyp-Konvertierungen
Informationen zu Konvertierung von Datenbank-Datentypen in XML-Schema-Datentypen finden Sie in den Anhängen.

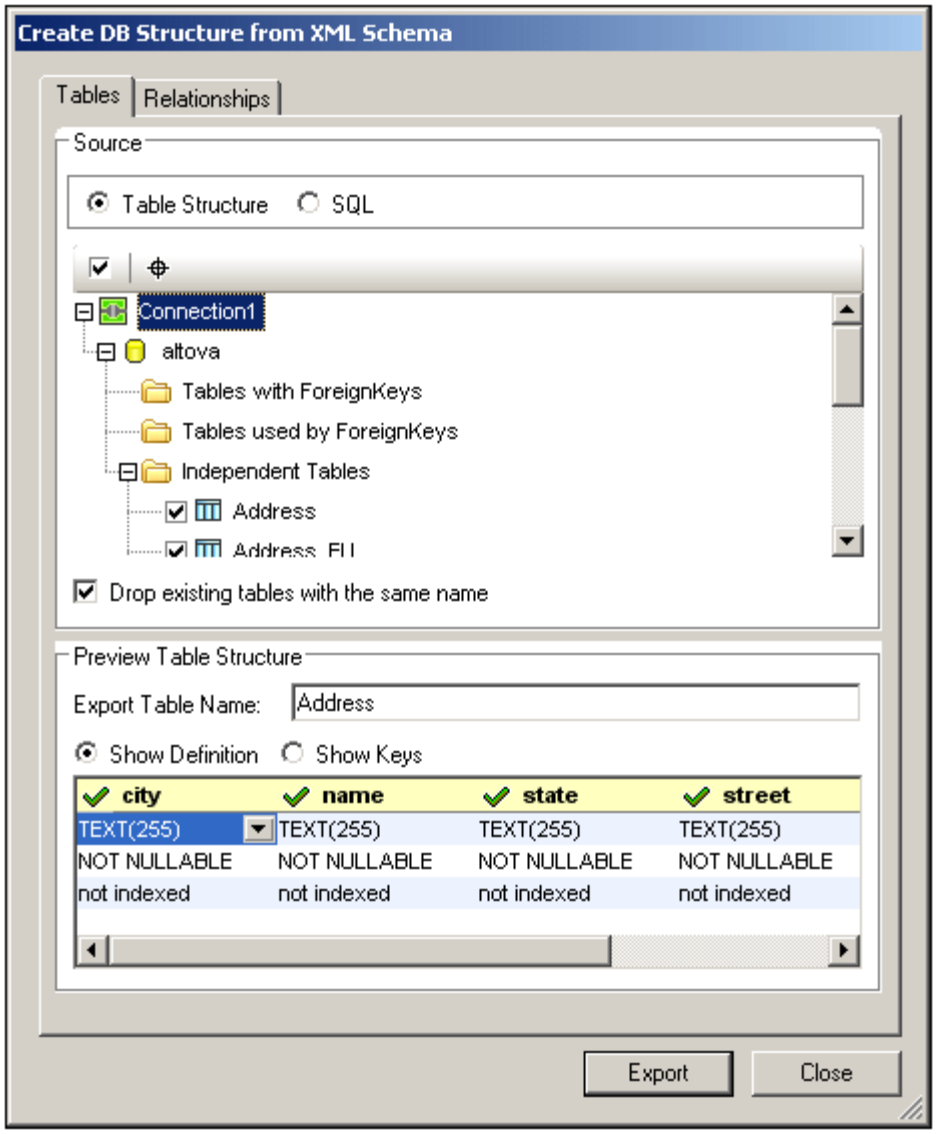
15.10.6 DB Struktur von XML-Schema erzeugen

In XMLSpy können Sie auf Basis einer vorhandenen Schema-Datei eine leere Datenbank (oder ein Datenbankgerüst) erstellen.

1. Öffnen Sie die Schema-Datei in der Schema/WSDL-Ansicht.
2. Wählen Sie den Menübefehl **Konvertieren | DB-Struktur von XML-Schema erzeugen**. Daraufhin wird das Dialogfeld Verbindung zur Datenquelle herstellen, in dem Sie eine Verbindung zu einer Datenbank (DB) herstellen können.
3. Gehen Sie vor, wie unter Herstellen einer Verbindung zu einer Datenquelle beschrieben, um eine Verbindung zur gewünschten Datenbank herzustellen. Um z.B. eine Verbindung zu einer Microsoft Access-Datenbank herzustellen, aktivieren Sie das Optionsfeld "Microsoft Access" und fahren Sie fort, um eine Datenbank auszuwählen. Sie können eine vorhandene Datenbank verwenden oder eine neue Datenbank erstellen , in der die Schemastruktur enthalten sein wird.
4. Im Dialogfeld "DB-Struktur von XML-Schema erzeugen" werden anhand des Schemas Tabellen erstellt und in hierarchischer Form an der Stelle in der Datebank angezeigt, wo sie in der DB vorkommen sollen. In der Abbildung unten wird z.B. die Tabelle "Address" erstellt und für den Export ausgewählt. Tabellen, die nicht exportiert werden sollen, müssen deselektiert werden (durch Deaktivieren des Kontrollkästchens oder

 © 2007 Altova GmbH

Auswahl des entsprechenden Eintrags für die Tabelle aus dem Kontextmenü).



5. Wenn Sie Tabellen in einer bestehenden DB löschen wollen (Drop), die denselben Namen wie Tabellen aus dem Schema haben, aktivieren Sie das Kontrollkästchen "Vorhandene Tabellen desselben Namens löschen".

	Direkt	über ODBC	über ADO
MS Access (2000 und 2003)	OK *	OK	OK
MS SQL Server	OK *	OK	OK
Oracle	OK *	OK	OK
MySQL	-	OK *	OK ⌖
Sybase	-	OK *	OK
IBM DB2	-	OK *	OK

* *Empfohlene Verbindungsmethode für die jeweilige Datenbank.*
⌖ *MySQL: Bei Herstellen der ADO-Verbindung auf Basis von ODBC wird empfohlen, entweder den Benutzer oder den System DSN zu verwenden.*
\- *Nicht unterstützt*

XMLSpy mappt sowohl hierarchisch als auch flach formatierte Schemas. XMLSpy erkennt beide Formate automatisch.
Das flache Format wird auf zwei verschiedene Arten auf SQL gemappt.

- SQL Server DB, Oracle DB oder Sybase DB:
  Bei einem Schema, das für eine der oben angeführten Datenbanken im flachen Format generiert wurde, wird der Schema-Katalogname extrahiert und im generierten SQL Script als DB-Name verwendet. Dies bedeutet, dass das erzeugte SQL Script an einer Ziel-DB ausgeführt wird, deren Name mit dem Schema-Katalognamen identisch sein muss.
- Access (2000 oder 2003), MySQL oder DB2 DB:
  Bei einem Schema, das für eine der oben angeführten Datenbanken im flachen Format generiert wurde, wird der Schema-Katalogname **ignoriert**, wenn das SQL Script generiert wird. Dies bedeutet, dass das erzeugte SQL Script an einer Ziel-DB ausgeführt wird, bei der der Benutzer eingeloggt war:

Datentyp-Konvertierungen
Informationen zu Konvertierung von Datenbank-Datentypen in XML-Schema-Datentypen finden Sie in den Anhängen.

15.10.7 Export in Textdateien

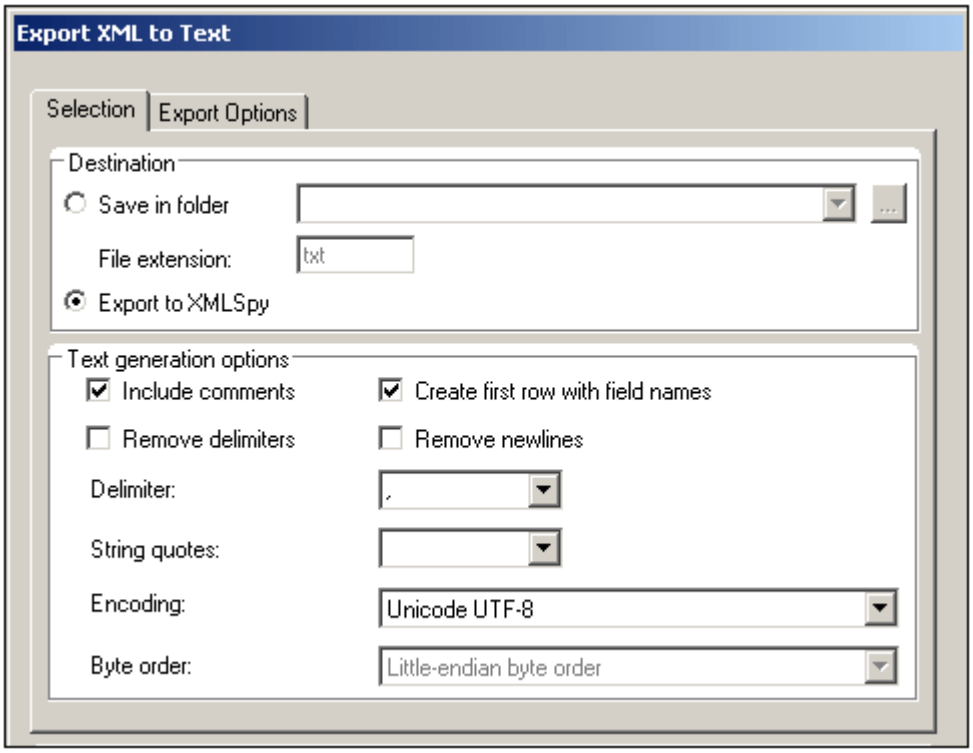
Mit dem Befehl **Konvertieren | Export in Textdateien** werden XML-Daten für den Austausch mit Datenbanken oder Altdatenbeständen in Textformate exportiert. Wenn Sie auf diesen Befehl klicken, wird das Dialogfeld "XML in Text exportieren" aufgerufen. Es besteht aus zwei Teilen (*die in der Abbildung unten separat angezeigt werden*):

- einem oberen Bereeich mit zwei Registern: (i) "Selection" und (ii) "Export Options".
- einem unteren Bereich, dem Vorschaufenster.

Nachdem Sie die gewünschten Optionen in diesem Dialogfeld ausgewählt haben (*Beschreibung siehe unten*), klicken Sie zum Exportieren der Textdatei(en) auf die Schaltfläche **Export**.

Selection
Auf dem Register "Selection" (*Abbildung unten*) können Sie den Zielordner für die zu exportierende Datei sowie die Textgenerierungsoptionen auswählen.

 © 2007 Altova GmbH



Ziel: Die exportierte Datei kann direkt in einem Ordner gespeichert werden. Die Dateierweiterung kann angegeben werden Die Dateinamen sind jene der Elemente (in der XMl-Datei), die exportiert werden. Alternativ dazu können auch Dateien ohne Titel in XMLSpy exportiert werden. Diese Dateien werden auf der Benutzeroberfläche angezeigt und können später gespeichert werden.

Comments inkludieren: Aktivieren Sie diese Option, um einen Kommentar in die exportierte XML-Datei zu inkludieren, der die zur Auswahl der Daten verwendete SQL-Abfrage sowie eine Liste mit einem Eintrag für jede Spaltenüberschrift in der Datenbanktabelle anzeigt.

Erste Zeile mit Feldnamen erzeugen: Ist diese Option aktiviert, enthalten die Tabellen die Namen der Datenbankspalten.

Trennzeichen entfernen: Entfernt die in Textwerten in den exportierten Daten enthaltenen Trennzeichen. Definieren Sie auf diesem Register in der Auswahlliste "Trennzeichen", welches Trennzeichen entfernt werden soll. Wenn diese Option z.B. aktiviert ist und das ausgewählte Trennzeichen das Apostroph ist, wird bei Export des XML-Werts `Ba'ker` der String im exportierten Text `Baker` lauten.

Neue Zeilen entfernen: Neue Zeilen werden aus den exportierten Daten entfernt.

Trennzeichen: Wählen Sie in der Dropdown-Liste das Zeichen aus, das beim Export entfernt werden soll. Geben Sie alternativ dazu den gewünschten Zeichenstring ein.

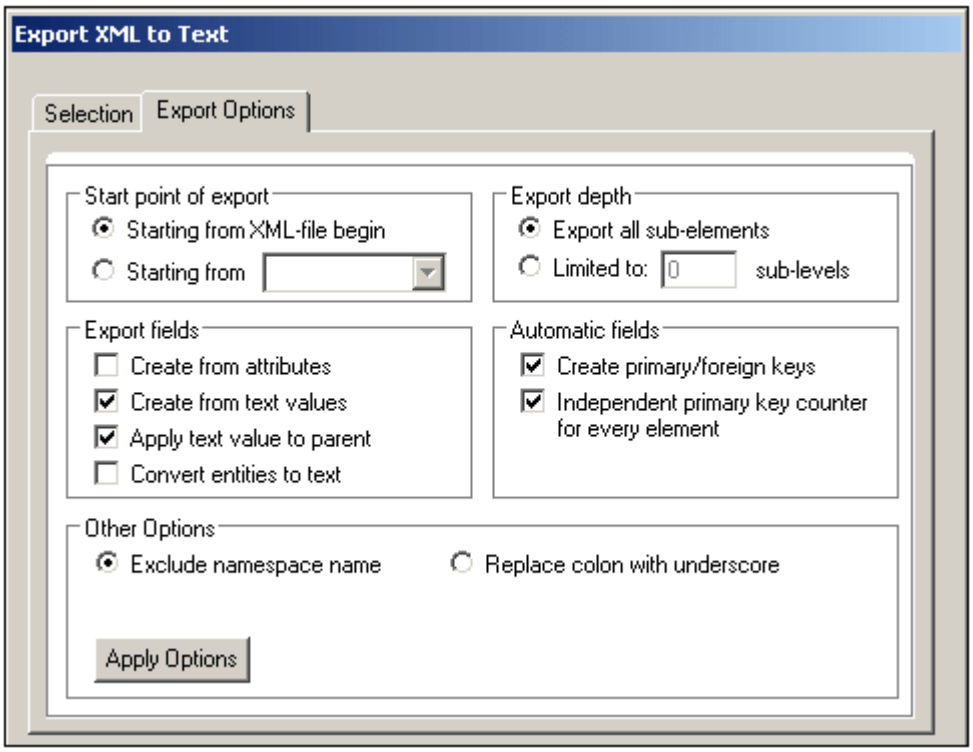
String-Anführungszeichen: In diese Fall wird jeder Datenstring in die in dieser Auswahlliste definierten Anführungszeichen gesetzt.

Encoding: Wählen Sie aus der Dropdown-Liste für die beim Export generierten Dateien das gewünschte Encoding aus.

Bytefolge: Bei Import von 16-Bit oder 32-Bit Unicode (UCS-2, UTF-16 oder UCS-4) Dateien

können Sie auch zwischen den Bytefolgen Little-Endian und Big-Endian wechseln.

Export Options
Weitere Export-Optionen (Beschreibung weiter unten) können auf dem Register "Export Options" (*Abbildung unten)* festgelegt werden.



Export beginnt: Sie können festlegen, ob das gesamte XML-Dokument exportiert werden soll oder Sie können den Export auf die Datenhierarchie, beginnend beim gerade ausgewählten Element beschränken. Die Anzahl der untergeordneten Ebenen unterhalb des zu exportierenden Startpunkts wird in der Option "Export-Tiefe" festgelegt.

Export-Tiefe: Gibt die Anzahl der untergeordneten Ebenen an, bis zu denen die Daten exportiert werden sollen.

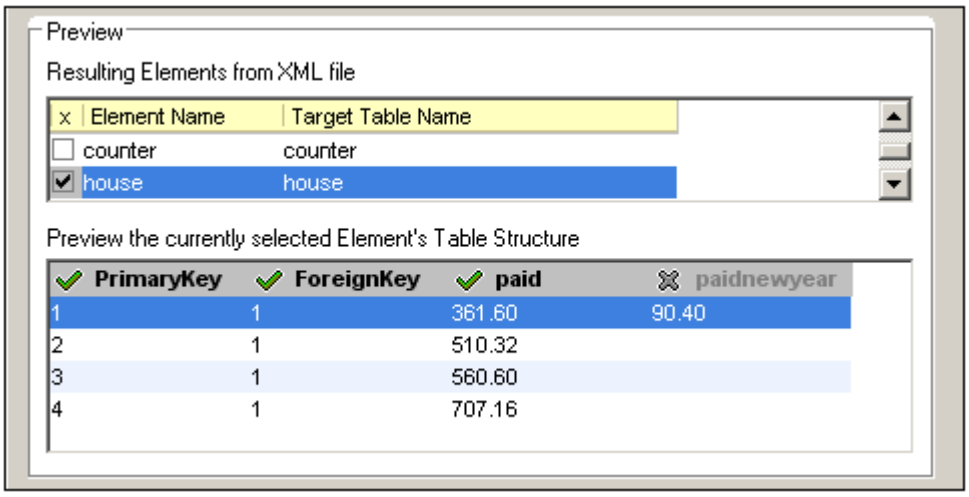
Exportfelder erzeugen aus: Abhängig von Ihren XML-Daten möchten Sie eventuell nur Elemente, Attribute oder den Textinhalt Ihrer Elemente exportieren. Beachten Sie, dass Sie den Export für einzelne Elemente im Vorschaufenster aufheben können.

Generiere automatische Felder: XMLSpy erzeugt für jeden ausgewählten Elementtyp eine Ausgabedatei oder -tabelle. Sie können festlegen, ob zum Verknüpfen Ihrer Daten im relevanten Modell automatisch Primär-/Sekundärschlüsselpaare erstellt werden sollen oder ob für jedes Element ein Primärschlüssel erstellt werden soll.

Namespace-Namen ausschließen: Zusammen mit dem Optionsfeld "Unterstrich ersetzt Doppelpunkt" bildet dies eine Entweder-Oder-Auswahl. Legt fest, ob Namespace-Präfixe von Elementen und Attributen ausgeschlossen werden sollen oder ob der Doppelpunkt im Namespace-Präfix durch einen Unterstrich ersetzt werden soll.

© 2007 Altova GmbH

Vorschaufenster
Das Vorschaufenster (*Abbildung unten)* wird unterhalb der Register "Selection" und "Export Options" angezeigt.



Im Bereich "Ergebniselemente aus der XML-Datei" wird der Node-Name angezeigt, der exportiert werden soll sowie der Name in der generierten Datei. Sie können auswählen, welche Nodes exportiert bzw. welche nicht exportiert werden sollen. Wenn ein Element ausgewählt wird, wird im zweiten Bereich unterhalb davon eine Vorschau seiner Struktur angezeigt. In diesem Fenster wird der Export der entsprechenden Spalte bei Klick links vom Spaltennamen aktiviert bzw. deaktiviert. In der Abbildung oben wurde die Spalte `paidnewyear` deaktiviert, sodass sie nicht exportiert wird.

15.10.8 Export in Datenbank

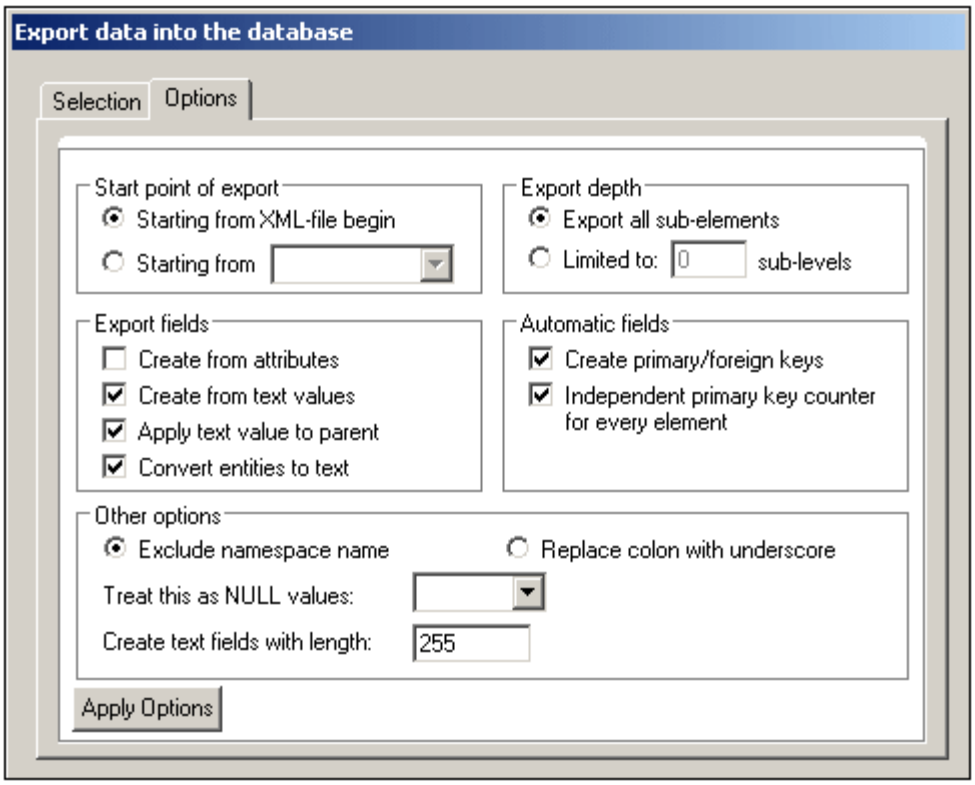
Mit dem Befehl **Konvertieren | Export in Datenbank** werden XML-Daten in eine Datenbank exportiert. Bei Auswahl dieses Befehls wird der Verbindungsassistent gestartet und gestattet Ihnen, eine Verbindung zu der Datenbank aufzubauen, die aktualisiert werden soll. Nach Herstellung der Verbindung wird das Dialogfeld "Daten in die Datenbank exportieren" angezeigt. Es besteht aus zwei Bereichen (*die in den Abbildungen unten separat gezeigt werden*):

- einem oberen Bereich mit zwei Registern: (i) Selection und (ii) Export Options.
- einem unteren Bereich, dem Vorschaufenster.

Nachdem Sie in diesem Dialogfeld (*Beschreibung siehe unten*) die gewünschten Optionen gewählt haben, klicken Sie auf die Schaltfläche **Exportieren**, um die Daten in die Datenbank zu exportieren.

Auswahl
Auf dem Register "Selection" können Sie die Zieldatenbank und die Tabellengenerierungsoptionen auswählen. Im Feld "Ziel" wird die Verbindung zur Datenbank ausgewählt. Sie müssen auswählen, ob die Daten als neue Tabellen erstellt werden sollen, ob bestehende Tabelle aktualisiert werden sollen oder ob zuerst versucht werden soll, eine bestehende Tabelle zu aktualisieren und anschließend eine neue Tabelle erstellt werden soll, falls das Aktualisieren nicht möglich ist. Außerdem können Sie auf Basis der Anzahl der Fehler eine Stopp-Aktion definieren und optional ein SQL Script Logging.

Exportoptionen
Auf dem Register "Export Options" (*Abbildung unten)* können Sie die unten beschriebenen Exportoptionen festlegen.



Export beginnt: Sie können festlegen, ob das gesamte XML-Dokument exportiert werden soll oder Sie können den Export auf die Datenhierarchie, beginnend beim gerade ausgewählten Element beschränken. Die Anzahl der untergeordneten Ebenen unterhalb des zu exportierenden Startpunkts wird in der Option "Export-Tiefe" festgelegt.

Export-Tiefe: Gibt die Anzahl der untergeordneten Ebenen an, bis zu denen die Daten exportiert werden sollen.

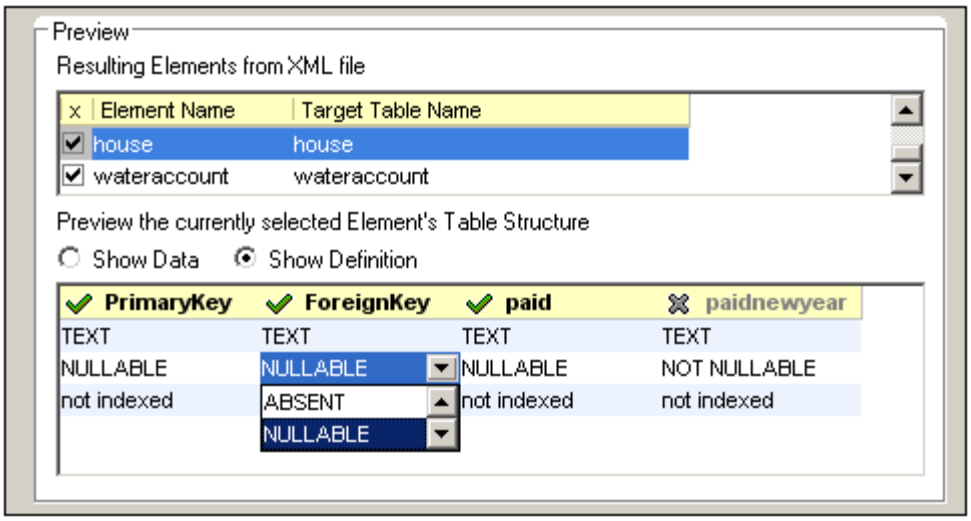
Exportfelder erzeugen aus: Abhängig von Ihren XML-Daten möchten Sie eventuell nur Elemente, Attribute oder den Textinhalt Ihrer Elemente exportieren. Beachten Sie, dass Sie den Export für einzelne Elemente im Vorschaufenster aufheben können.

Generiere automatische Felder: XMLSpy erzeugt für jeden ausgewählten Elementtyp eine Ausgabedatei oder -tabelle. Sie können festlegen, ob zum Verknüpfen Ihrer Daten im relevanten Modell automatisch Primär-/Sekundärschlüsselpaare erstellt werden sollen oder ob für jedes Element ein Primärschlüssel erstellt werden soll.

Namespace-Namen ausschließen: Zusammen mit dem Optionsfeld "Unterstrich ersetzt Doppelpunkt" bildet dies eine Entweder-Oder-Auswahl. Legt fest, ob Namespace-Präfixe von Elementen und Attributen ausgeschlossen werden sollen oder ob der Doppelpunkt im Namespace-Präfix durch einen Unterstrich ersetzt werden soll.
Null-Wert und Textfeldlänge: Sie können angeben, welche Textstrings als NULL-Werte behandelt werden sollen. Außerdem können Sie die Länge von Textfeldern in der Datenbank definieren.

© 2007 Altova GmbH

Vorschaufenster
Das Vorschaufenster (*Abbildung unten)* wird unterhalb der Register "Selection" und "Export Options" angezeigt.



Im Bereich "Ergebniselemente aus der XML-Datei" wird der Node-Name angezeigt, der exportiert werden soll sowie der Name in der generierten Datei. Sie können auswählen, welche Nodes exportiert bzw. welche nicht exportiert werden sollen. Wenn ein Element ausgewählt wird, wird im zweiten Bereich unterhalb davon eine Vorschau seiner Struktur angezeigt. In diesem Fenster wird der Export der entsprechenden Spalte bei Klick links vom Spaltennamen aktiviert bzw. deaktiviert. In der Abbildung oben wurde die Spalte `paidnewyear` deaktiviert, sodass sie nicht exportiert wird.

Wenn in der Elementtabelle die Daten angezeigt werden, kann der Key Constraint-Typ durch Klicken links von einem Spaltennamen ausgewählt werden. Wenn in der Struktur der Elementtabelle Felddefinitionen angezeigt werden sollen, können Sie die Definitionen bearbeiten, indem Sie die Definition auswählen und eine Option aus der Auswahlliste der Definition auswählen (*siehe Abbildung oben*).

15.11 Menü "Ansicht"

Im Menü Ansicht kontrollieren Sie die Anzeige für das aktive Hauptfenster. So können Sie sich in XMLSpy Ihre XML-Dokumente verschiedenartig anzeigen lassen.

In diesem Abschnitt finden Sie eine vollständige Beschreibung aller Befehle im Menü "Ansicht".

15.11.1 Text-Ansicht

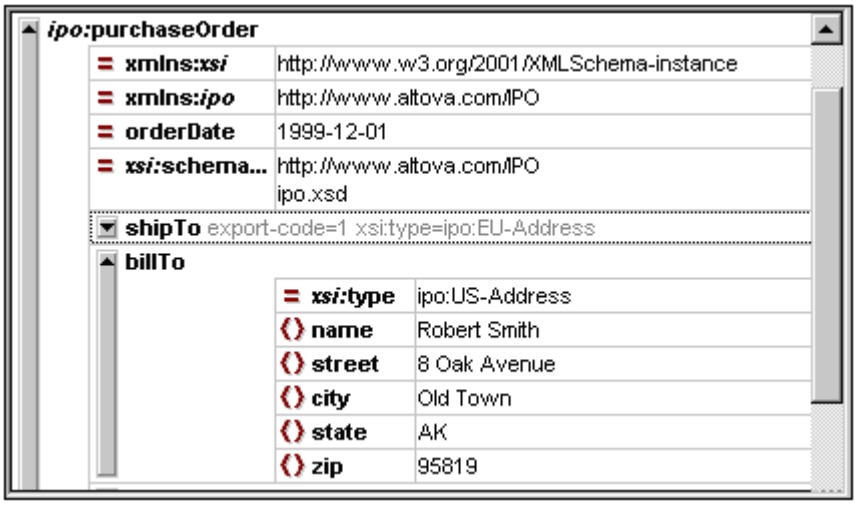
Mit diesem Befehl **wechseln** Sie für das aktuelle Dokument in die Text-Ansicht. In dieser Ansicht können Sie das Dokument in der Textform editieren. Die Text-Ansicht unterstützt eine Reihe von Editierfunktionen, die im Abschnit Text-Ansicht näher beschrieben sind.

Bitte beachten Sie: Sie können Aspekte der Textansicht mit Hilfe von Optionen konfigurieren, die auf den verschiedenen Registern des Dialogfelds "Optionen" (**Extras | Optionen**) zur Verfügung stehen.

15.11.2 Enhanced Grid Ansicht

Mit diesem Befehl **wechseln** Sie mit dem aktuellen Dokument in die Enhanced Grid Ansicht.

Wenn Sie von der Text-Ansicht in diese Ansicht wechseln, wird Ihr Dokument vorab auf Wohlgeformtheit überprüft.



Eingebettete Datenbank/Tabelle Ansicht
Mit XMLSpy können Sie sich wiederholende Elemente innerhalb der Enhanced Grid Ansicht in einer Datenbank/Tabelle Ansicht ansehen. Diese Funktion ist immer dann verfügbar, wenn Sie die Enhanced Grid Ansicht aktivieren können, und kann beim Editieren jeder beliebigen XML Datei - XML, XSD, XSL usw. verwendet werden.

Weitere Informationen zu diesem Thema finden Sie auch im Abschnitt Grid-Ansicht.

© 2007 Altova GmbH

15.11.3 Schema/ WSDL Design-Ansicht

Mit diesem Befehl wechseln Sie mit dem aktuellen Dokument in die Schema/WSDL Design-Ansicht. Nähere Informationen dazu erhalten Sie im Abschnitt Schema/WSDL Design-Ansicht.

15.11.4 Authentic-Ansicht

Mit diesem Befehl wechseln Sie mit Ihrem Dokument in die Authentic-Ansicht.

In der Authentic-Ansicht können Sie XML-Dokumente bearbeiten, die auf in **<%XDESIGN%> erstellten StyleVision Power Stylesheet-Vorlagen basieren!** Die Vorlagen werden in <%XDESIGN%> als StyleVision Power Stylesheet (***.sps** Dateien) gespeichert, und liefern alle für die Authentic-Ansicht benötigten Informationen.

Öffnen Sie die Vorlagen über den Menübefehl **Datei | Neu** und klicken auf die Schaltfläche "StyleVision Stylesheet auswählen...". Weitere Informationen finden Sie in der Dokumentation zur Authentic-Ansicht.

Bitte beachten Sie: Wenn bei Umschalten in die Authentic-Ansicht eine Meldung angezeigt wird, dass eine temporäre Datei nicht erstellt werden konnte, wenden Sie sich an Ihren Systemadministrator, damit dieser die Standard-Sicherheits-ID für Personen, die nicht als Hauptbenutzer angemeldet sind, ändert, so dass diese Ordner und Dateien erstellen können.

15.11.5 Browser-Ansicht

Mit diesem Befehl wechseln Sie mit dem aktuellen Dokument in die Browser Ansicht.

Diese Ansicht verwendet einen XML-fähigen Browser um das XML-Dokument mit den aus CSS oder XSL Style-Sheets zur Verfügung gestellten Informationen darzustellen.

Wenn Sie in die Browser Ansicht wechseln, wird das Dokument zuerst auf Gültigkeit geprüft, wenn Sie die Option "Validieren beim Speichern der Datei" unter **Extras | Optionen** auf dem Register **Datei** einschalten.

Weitere Informationen zu dieser Ansicht finden Sie im Abschnitt Hauptfenster.

15.11.6 Erweitern

Kürzel: **"+" auf dem Ziffernblock**

Mit diesem Befehl blenden Sie eine weitere Ebene des ausgewählten Elements ein.

Der Befehl kann in der Enhanced Grid Ansicht ausgeführt werden.

In der Enhanced Grid Ansicht, bleiben das Element und seine Child-Elemente nach dem Einblenden markiert. So können Sie große Elemente einblenden indem Sie wiederholt die + Taste betätigen.

Sie können Elemente auch durch Klicken auf die graue Leiste links eines jeden Elements einblenden oder ausblenden.

15.11.7 Reduzieren

Kürzel: **"-" auf dem Ziffernblock**

Mit diesem Befehl **blenden** Sie ein Element um eine Ebene **aus**.

Der Befehl kann in der Enhanced Grid Ansicht ausgeführt werden.

Sie können Elemente auch durch Klicken auf die graue Leiste links eines jeden Elements einblenden oder ausblenden.

15.11.8 Alles erweitern

Kürzel: **"*" auf dem Ziffernblock**

Mit diesem Befehl **blenden** Sie alle Child-Objekte des **ausgewählten Elements** ein.

Der Befehl kann in der Enhanced Grid Ansicht ausgeführt werden.

15.11.9 Unselektierte reduzieren

Kürzel: **CTRL** + **"-" auf dem Ziffernblock**

Mit diesem Befehl können Sie sich auf ein Element und dessen Children konzentrieren und alle anderen Elemente ignorieren.

Der Befehl kann in der Enhanced Grid Ansicht ausgeführt werden.

Wählen Sie das Element aus, mit dem Sie arbeiten möchten und blenden Sie mit diesem Befehl alle anderen nicht selektierten Elemente aus.

15.11.10 Optimale Breite

Mit diesem Befehl stellen Sie die Größe der Spalten so ein, dass der Text des gesamten Dokuments in die zugehörigen Spalten passt.

Wenn Sie mehrere Elemente ein- oder ausblenden, wählen Sie diesen Befehl, da für die Berechnung der optimalen Breite nur sichtbare Objekte in Betracht gezogen werden.

15.11.11 Zeilenumbruch

Mit diesem Befehl schalten Sie den Zeilenumbruch in der **Text-Ansicht** ein oder aus.

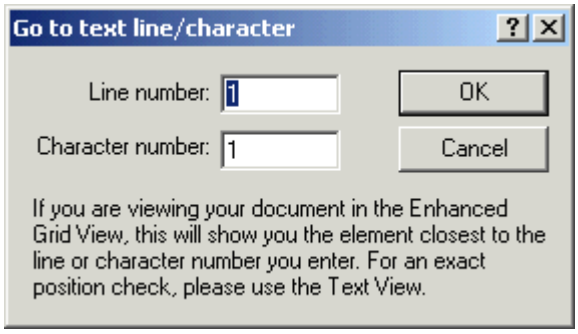
© 2007 Altova GmbH

15.11.12Gehe zu Zeile/Zeichen

Kürzel: **Strg+g**

Mit diesem Befehl gelangen Sie zu einer **spezifischen Zeilennummer** und/oder Zeichenposition in einem XML-Dokument in der Text-Ansicht.

Wenn Sie mit einem externen XSLT Prozessor (weitere Infos: Register XSL unter Extras | Optionen) arbeiten, werden Sie öfter Fehlermeldungen bezüglich Zeilennummer und/oder Zeichenposition erhalten. Mit XMLSpy können Sie mit diesem Befehl exakt zur gewünschten Position navigieren.



Diese Funktion ist in der Text-Ansicht und Enhanced Grid Ansicht verfügbar, zeigt allerdings in der Grid-Ansicht nur die ungefähre Position an, indem das Element markiert wird, welches sich am nächsten zur gewünschten Position befindet.

15.11.13Gehe zu Datei

Mit diesem Befehl **öffnen** Sie das Dokument, welches auf das von Ihnen momentan bearbeitete Dokument **referenziert**.

Wählen Sie den gewünschten Dateinamen, Pfadnamen, oder die URL und führen Sie diesen Befehl über das Menü Ansicht aus.

Sie können folgendes auswählen:

- Ein vollständiges Element oder Attribut in der Enhanced Grid Ansicht
- Zeichen innerhalb eines beliebigen Elements in der Text oder Enhanced Grid -Ansicht.
- Einen kompletten String. Wenn sich der Cursor zwischen zwei Anführungszeichen befindet, benutzt XMLSpy automatisch den kompletten String der sich zwischen den Anführungszeichen befindet.

15.11.14Zeilennummerierungsleiste

Mit diesem Befehl wird die Zeilennummerierungsleiste in der Text-Ansicht **ein- oder ausgeblendet**.

15.11.15Lesezeichenleiste

Mit diesem Befehl wird die Lesezeichenleiste in der Text-Ansicht **ein- oder ausgeblendet**. Wenn in der Lesezeichenleiste ein Lesezeichen vorhanden ist und die Lesenzeichenleiste anschließend ausgeblendet wird, erscheint die mit dem Lesezeichen versehene Zeile markiert. Nach Einblenden der Lesenzeichenleiste wird die Markierung entfernt und Sie sehen darin das Lesezeichen.

15.11.16Klappleiste

Mit diesem Befehl blenden Sie die Klappleiste in der Textansicht ein bzw. aus. Wenn Sie auf der Leiste auf das Plus- bzw. Minus-Symbol klicken wird der entsprechende Code-Abschnitt erweitert (ausgeklappt) bzw. reduziert (eingeklappt).

Bitte beachten Sie:
Sie können Child-Nodes in der Text-Ansicht erweitern oder reduzieren:

Umschalt+Klick auf ein Plus-Symbol **reduziert** alle Child-Nodes dieses Node, während

Strg+Klick auf einen zuvor reduzierten Node alle Child-Nodes **erweitert.**

15.11.17Einrücklinien

Mit diesem Befehl werden die Einrücklinien in der Text-Ansicht ein- oder ausgeblendet. Die Einrücklinien sind vertikale Linien, die entlang des Leerzeichenrandes verlaufen und die hierarchische Ebenenstruktur grafisch verdeutlichen.

© 2007 Altova GmbH

15.12 Menü "Browser"

Dieses Menü enthält Befehle, die nur verfügbar sind, wenn die Browser Ansicht aktiviert ist.

15.12.1 Zurück

Taste **Zurück**

Mit diesem Befehl wird eine zuvor angezeigte Seite aufgerufen. Mit der Zurück-Taste auf Ihrer Tastatur erreichen Sie denselben Effekt.
Dieser Befehl ist hilfreich, wenn Sie auf einen Link in Ihrem XML-Dokument klicken und danach zu Ihrem XML-Dokument zurückkehren möchten.

15.12.2 Vorwärts

Dieser Befehl ist nur dann verfügbar, wenn Sie zuvor den Befehl **Zurück** verwendet haben. Mit diesem Befehl gelangen sie vorwärts zur vorher angezeigten Seite.

15.12.3 Abbrechen

Mit diesem Befehl beenden Sie das Laden eines neuen Dokuments.
Dieser Befehl ist nützlich, wenn Sie große externe Dateien oder graphische Darstellungen über eine langsame Internetverbindung laden.

15.12.4 Aktualisieren

F5

Mit diesem Befehl aktualisieren Sie die Browser Ansicht durch erneutes Laden des XML-Dokuments, sowie potentieller externer CSS oder XSL Stylesheets und DTDs.

15.12.5 Schriftgröße

Mit diesem Befehl können Sie die Schriftgröße wählen, die für Text in Ihrem XML-Dokument verwendet werden soll. Die Funktionsweise ist ähnlich wie die, welche in den meisten Browsern zur Anwendung kommt.

15.12.6 Neues Fenster

Mit diesem Befehl öffnen Sie die Browser Ansicht in einem neuen Fenster, sodass Sie Dokumente nebeneinander ansehen können.
Wenn Sie die Browser Ansicht in einem neuen Fenster geöffnet haben, drücken Sie in der Bearbeitungsansicht die Taste **F5**, um die Browser-Ansicht automatisch zu aktualisieren. Um separate Fenster wieder anzudocken, klicken Sie auf die Maximieren-Schaltfläche in der rechten oberen Ecke des aktiven Fensters.

15.13 Menü "WSDL"

Dieses Menü enthält Befehle, die nur zur Verfügung stehen, wenn Sie ein WSDL-Dokument in der Schema/WSDL-Ansicht anzeigen.

Weitere Informationen finden Sie im WSDL Tutorial und in der Dokumentation zur WSDL Design-Ansicht.

15.13.1 Messages

Message einfügen
Eine neue Message wird in der WSDL Datei eingefügt. In der Eingabehilfe "Übersicht" wird der Message-Eintrag geöffnet und die neu erstellte Message erscheint dort markiert.

Message löschen
Löscht die ausgewählte Message aus dem Input- oder Outputelement.

Message Part (Parameter) einfügen
Ein Message Part (Parameter) wird zur ausgewählten Nachricht hinzugefügt.

Message Part (Parameter) löschen
Ein Message Part (Parameter) wird aus der ausgewählten Nachricht gelöscht.

15.13.2 Operations

Operation einfügen
Eine neue Operation wird in das Operationsfenster hinzugefügt.

Operation löschen
Die ausgewählte Operation wird aus dem Operationsfenster gelöscht.

Input-Element einfügen
Ein Input-Element wird in der ausgewählten Operation hinzugefügt.

Output-Element einfügen
Ein Output-Element wird in der ausgewählten Operation hinzugefügt.

Fault-Element einfügen
Ein Fault-Element wird in der ausgewählten Operation hinzugefügt.

Input/Output/Fault-Element löschen
Löscht das ausgewählte Input-, Output- oder Fault-Element.

Neue Message zu Input/Output/Fault-Element hinzufügen
Eine neue (Default) Message wird zum ausgewählten **Input-** oder **Output-**Element hinzugefügt.

Leere Operation
Fügte eine leere Operation zum derzeit ausgewählten PortType hinzu.

15.13.3 PortType

PortType einfügen
Ein neuer Porttyp wird im Porttyp-Fenster hinzugefügt.

© 2007 Altova GmbH

PortType löschen
Der ausgewählte Porttyp wird aus dem Porttyp-Fenster gelöscht.

15.13.4 Binding

Neues Binding
Ein neues Binding wird im Bindings-Fenster hinzugefügt.

Binding löschen
Das ausgewählte Binding wird aus dem Bindings-Fenster gelöscht.

Child anhängen
Ein neues Extensibility-Element wird an eine Input- oder Output Message angehängt. Falls ein Menüpunkt an der ausgewählten Stelle nicht angehängt werden darf, ist dieser ausgegraut. Weitere Informationen zu Extensibility-Objekten finden Sie in der W3C WSDL Spezifikation.

Folgende Extensibility-Objekte stehen zur Verfügung:

- soap:body
- soap:header
- soap:headerfault
- soap:fault
- mime:content
- mime:multipartrelated
- mime:part
- mime:mimeXml
- http:urlencoded
- http:urlreplacement

Extensibility löschen
Löscht das markierte Extensibility-Objekt.

15.13.5 Service

Service einfügen
Ein neuer Service wird im Services-Fenster hinzugefügt.

Service löschen
Der ausgewählte Service wird aus dem Services-Fenster gelöscht.

Port einfügen
Ein neuer Port wird dem ausgewählten Service im Services-Fenster hinzugefügt.

Port löschen
Der ausgewählte Port wird aus dem ausgewählten Service gelöscht.

15.13.6 Types

Neues Schema
Diese Option ist dann verfügbar, wenn die WSDL Datei kein Schema-Element enthält. Bitte beachten Sie, dass bei Aufrufen der Menüoption **Datei | Neu** ein Schemaelement im Skelett der WSDL Datei inkludiert ist. In diesem Fall ist dieser Menüpunkt nicht verfügbar (siehe unten).

```
<types>
        <xs:Schema/>
</types>
```

Schema(s) in Schema-Ansicht bearbeiten
Öffnet eine Schemaskelett-Datei, falls die WSDL Datei keine Referenz auf ein spezifisches Schema enthält. Dies ist der Fall, wenn Sie die Menüoption **Datei | Neu** verwendet haben.

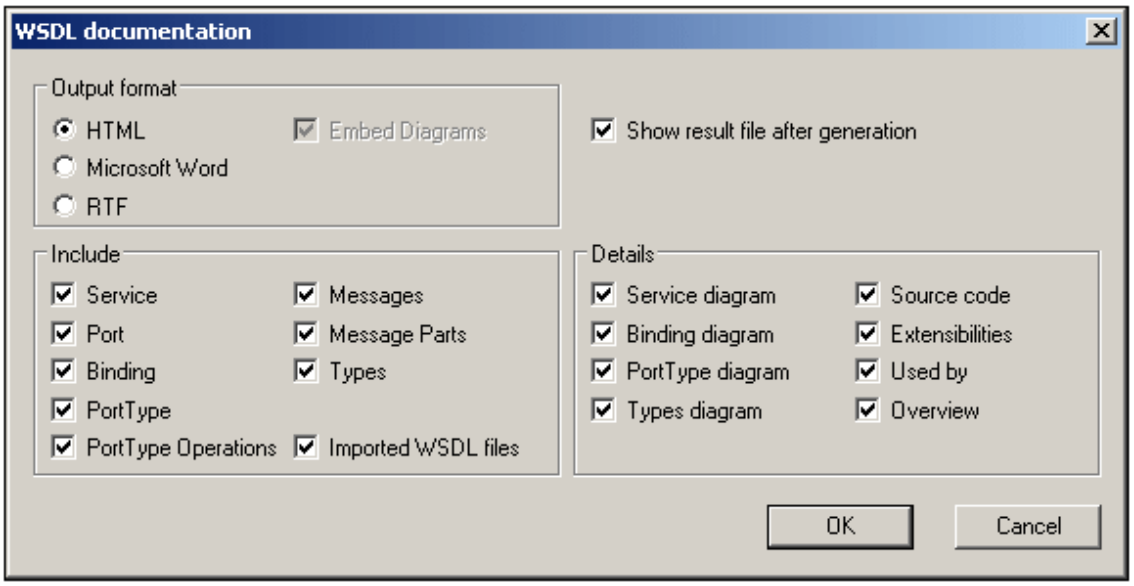
Wenn eine Referenz zu einem spezifischen Schema existiert, wird das Schema in der eingebetteten Schema-Ansicht des graphischen WSDL Editors angezeigt.

15.13.7 Diagramm speichern, Dokumentation generieren

Mit dem Befehl **Diagramm speichern** wird das Design-Diagramm als PNG-Datei gespeichert.

Mit dem Befehl **Dokumentation generieren** wird die Dokumentation für die aktuelle WSDL Datei generiert. Die Ausgabe der Dokumentation kann als HTML-, MS Word oder RTF-Datei erfolgen. Die Dokumentation wird für die Komponenten generiert, die Sie im Dialogfeld "WSDL-Dokumentation generieren" auswählen (Aufruf des Dialogfelds mit dem Befehl "Dokumentation generieren"). Miteinander in Zusammenhang stehende Komponenten werden in der Ausgabe auf dem Bildschirm durch Hyperlinks verbunden, um die Navigation zu vereinfachen. Beachten Sie, dass Dokumentation auch für **importierte WSDL-Dateien** generiert wird.

Anmerkung: Um Dokumentation im MS Word-Format generieren zu können, muss MS Word (Version 2000 oder höher) installiert sein.



Bei Aufruf des Befehls **Dokumentation generieren** wird das Dialogfeld **WSDL--Dokumentation** geöffnet:

- Das gewünschte Format wird im Bereich "Output-Format" definiert: entweder HTML, Microsoft Word oder RTF. Die Dokumentation kann entweder in Form einer einzigen Datei generiert oder auf mehrere Dateien aufgeteilt werden. Wenn mehrere Dateien generiert werden, entspricht jede Datei einer Komponente. Welche Komponenten in der Ausgabe inkludiert werden sollen, wird mit Hilfe der Kontrollkästchen im Bereich "Inkludieren" definiert.
- Die Option "Diagramme einbetten" ist bei den Ausgabeoptionen MS Word und RTF

aktiv. Wenn diese Option aktiviert ist, werden Diagramme entweder im PNG- oder im EMF-Format in die Ergebnisdatei eingebettet. Andernfalls werden Diagramme als PNG- oder EMF-Diagramme erstellt, welche in der Ergebnisdatei über Objektlinks angezeigt werden. Bei Ausgabe von HTML werden alle Diagramme als dokumentexterne PNG-Dateien erstellt.

- Im Fenster "Inkludieren" können Sie auswählen, welche Elemente in der Dokumentation berücksichtigt werden sollen. Bei Auswahl der Übersichtsoption werden alle Komponenten nach Komponententyp geordnet am Anfang der Datei angezeigt.
- Im Fenster "Details" werden die Einzelheiten aufgelistet, die bei den einzelnen Komponenten inkludiert werden können. Wählen Sie aus, welche Details in der Dokumentation berücksichtigt werden sollen.
- Die Option "Ergebnisdatei anzeigen" steht für alle drei Ausgabeoptionen zur Verfügung. Wenn diese Option aktiviert ist, werden die Ergebnisdateien in der Browser-Ansicht (HTML-Ausgabe), in MS Word (MS Word-Ausgabe) und der Standardapplikation für .rtf-Dateien (RTF-Ausgabe) angezeigt.

15.13.8 WSDL-Dokument neu parsen

Parst das Dokument neu. Dies ist in einigen Situationen erforderlich, z.B. wenn ein mit dem WSDL-Dokument verknüpftes Schema geändert wurde. Wenn Sie das WSDL-Dokument neu parsen, wird es mit den Informationen aus dem geänderten Schema-Dokument aktualisiert.

15.14 Menü "Soap"

XMLSpy unterstützt SOAP Version 1.1 und WSDL Version 1.1.

Bitte beachten Sie, dass bei der Erzeugung von SOAP-Requests das sichere http-Protokoll **HTTPS** nicht unterstützt wird.

Im Abschnitt **Soap Debugger**, der auf die Menübeschreibungen folgt, wird anhand des **nanonull.com timeservice Server**, der Ihnen von Altova zur Verfügung gestellt wird, die Funktionsweise des SOAP Debuggers beschrieben. Bitte verwenden Sie diesen Service, um den Debugger zu testen. Der AirportWeather Webservice, der auf den folgenden Seiten beschrieben wird, steht möglicherweise nicht immer zur Verfügung.

Verwenden Sie die SOAP Funktionalitäten:

- zum Testen Ihrer Webservices, ohne dass Client-Applikationen implementiert werden müssen
- zum schnellen Testen von Webservices von anderen Anbietern

Weitere Informationen zu den Spezifikationen erhalten Sie unter:
SOAP http://www.w3.org/TR/SOAP/
WSDL http://www.w3.org/TR/wsdl

15.14.1 SOAP Request erzeugen

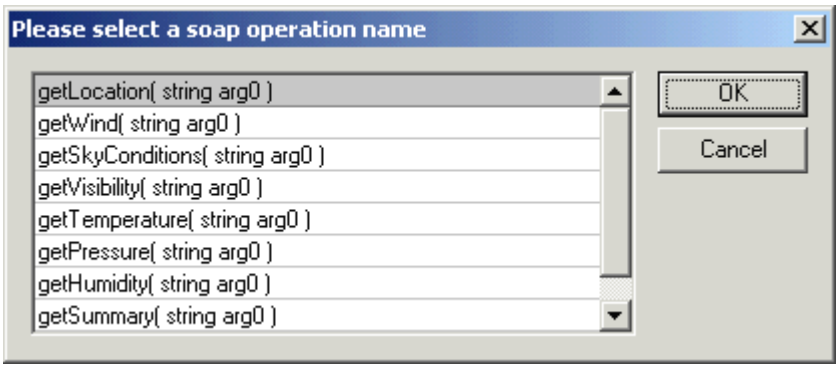
Mit diesem Befehl erzeugen Sie ein neues SOAP Request-Dokument.

Erzeugen eines neuen SOAP Requests:

1. Wählen Sie die Menüoption **Soap | SOAP Request erzeugen**.
   Im Dialogfeld geben Sie den lokalen Pfad oder die URL einer WSDL Datei ein, die den Webservice beschreibt. In diesem Beispiel verwenden Sie bitte capescience.com AirportWeather Webservice. Die URL für diesen Webservice lautet:
   http://www.capescience.com/webservices/airportweather/index.shtml

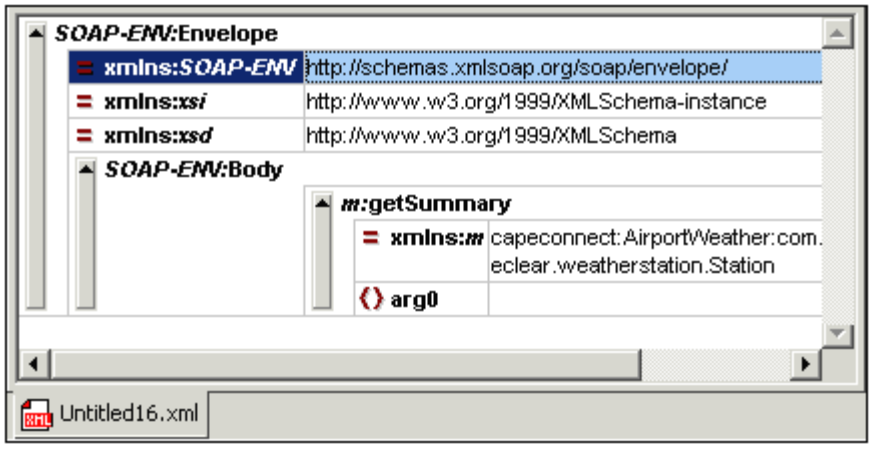
2. Klicken Sie auf OK um zu bestätigen und zum Herstellen der Verbindung zum capescience.com Server.
   Ein Dialogfeld wird angezeigt, in dem Sie eine bestimmte SOAP Operation auswählen können.

© 2007 Altova GmbH

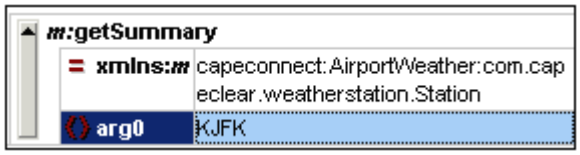


Bitte beachten Sie:
Sollte die Verbindung nicht zustande kommen, oder die URL nicht korrekt sein, wird an dieser Stelle eine Fehlermeldung angezeigt.

3. Klicken Sie auf den Eintrag **getSummary (string arg0)** und bestätigen Sie mit OK. Ein Untitled.XML-Dokument wird in XMLSpy geöffnet, welches die SOAP-Request Daten enthält.



4. Klicken Sie auf eines der Felder zum Abwählen der Markierung der Tabelle. Bevor der Request abgesendet werden kann, müssen Sie angeben, für welchen Flugplatz Sie eine Wettervorhersage benötigen.
5. Klicken Sie auf das Feld **arg0** und geben Sie **KJFK** für den Flughafen New York JFK Airport ein.

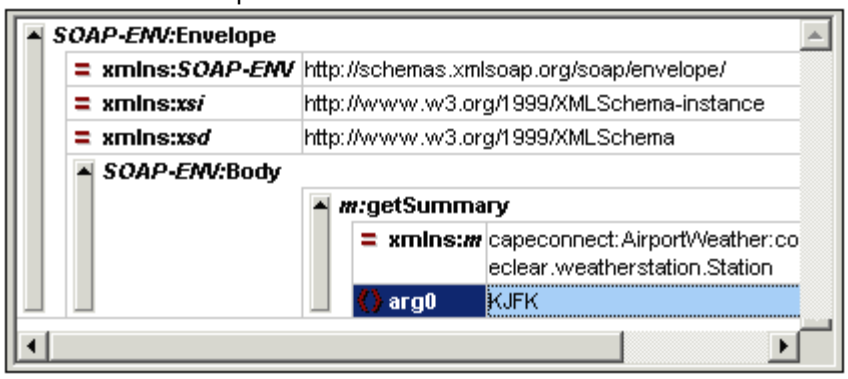


Nun ist die **Definition** des SOAP Requests vollständig und wird versendet. Jetzt müssen wir nur noch warten, ob wir eine Antwort vom Server erhalten. Siehe auch "Request an Server senden".

15.14.2 SOAP Request an Server senden

Mit diesem Befehl wird der SOAP Request über XMLSpy an den Webservice gesendet. Wenn Sie ein SOAP Request Dokument erzeugt haben, (siehe hierzu Neuen SOAP Request erzeugen), wird der Request über diesen Befehl versendet.

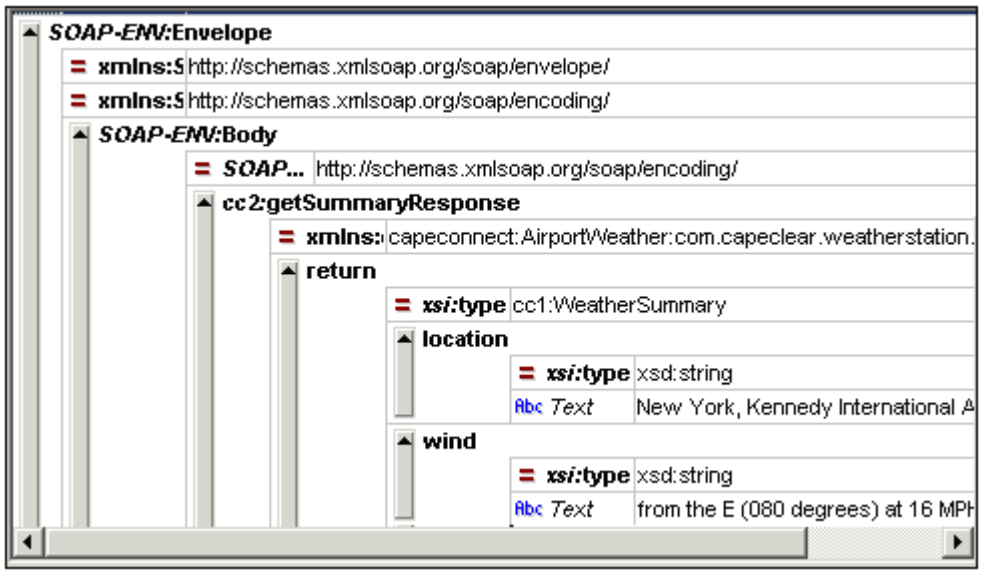
Zuvor definiertes SOAP Request Dokument:



So versenden Sie einen SOAP Request:

1. Stellen Sie sicher, dass das SOAP Request Dokument aktiv ist, und wählen Sie die Menüoption **Soap | SOAP Request an Server senden**.

   Nach einigen Sekunden wird automatisch ein neues XML-Dokument erstellt, welches den SOAP Response vom zuständigen Server enthält. In diesem Beispiel werden im neuen Dokument Standort und Wetterinformationen für den Flughafen Neu York JFK dargestellt.



Um die SOAP Request Parameter einzustellen, die nicht im Request Dokument angezeigt werden (da diese im HTTP Header transferiert werden), verweisen wir auf "SOAP Request Parameter ändern".

Speichern und erneute Verwendung eines SOAP Request:
In XMLSpy können Sie einen SOAP Request speichern und zu einem späteren Zeitpunkt erneut senden:

Nach Erzeugen eines SOAP Request mittels der Option **SOAP Request erzeugen**

1. Wählen Sie die Menüoption **Datei | Speichern unter...** und benennen Sie das SOAP Request XML-Dokument.
   Schließen Sie die SOAP Request-Datei.
2. Öffnen Sie das SOAP Request XML-Dokument erneut und wählen Sie die Menüoption **SOAP | SOAP Request-Parameter ändern.**
3. Daraufhin werden Sie gefragt, ob Sie diese Datei als SOAP Request definieren wollen

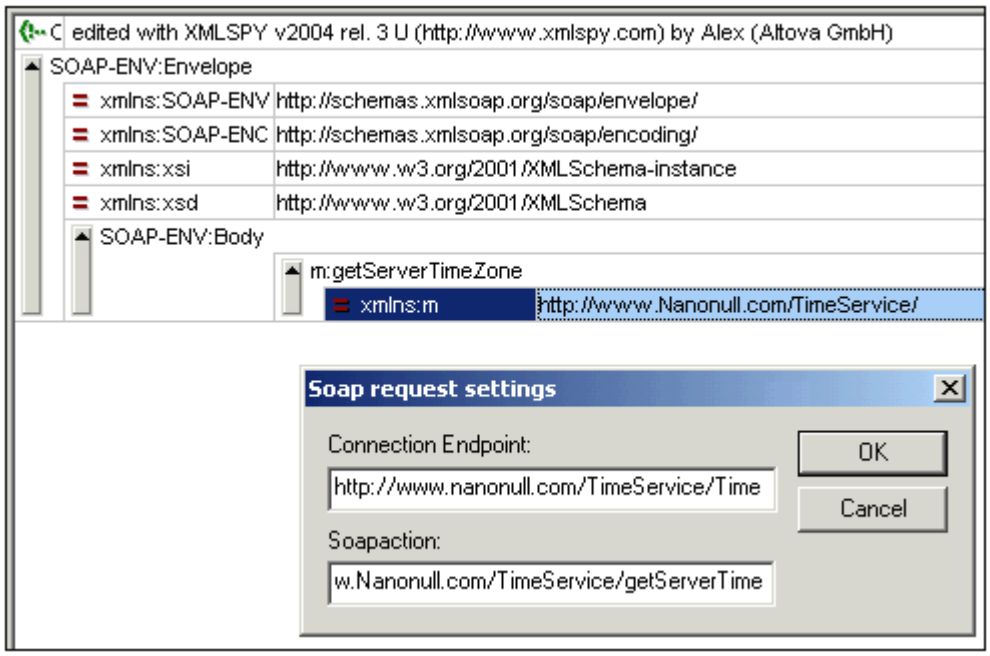
 © 2007 Altova GmbH

und informiert, dass Sie die SOAP Request Parameter manuell eingeben müssen. (Als SOAP Request-Dokument kann jede XML-Datei verwendet werden).

4. Klicken Sie auf **Ja** und geben Sie den **Connection Endpoint** und die **SOAP Operation** in die entsprechenden Felder ein.
5. Wählen Sie die Menüoption **SOAP | SOAP Request an Server senden,** um den Request abzusenden.

Bitte beachten Sie:

Diese beiden Parameter **werden nicht gespeichert**, wenn Sie das SOAP Request-Dokument speichern. Sie müssen sie manuell eingeben, wenn Sie das SOAP Request-Dokument erneut öffnen.

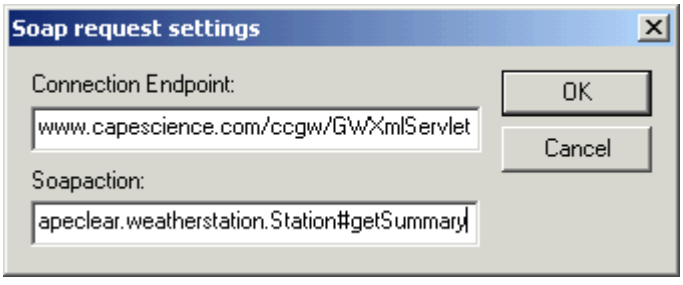


15.14.3 SOAP Request-Parameter ändern

Mit diesem Befehl können Sie die aktuell gültigen SOAP Parameter des SOAP Request Dokuments ändern. Das vorliegende Beispiel verwendet das zuvor definierte SOAP Request Dokument.

So ändern Sie die Parameter für den SOAP Request:

1. Stellen Sie sicher, dass das SOAP Request Dokument aktiv ist und wählen Sie die Menüoption **Soap | SOAP Request-Parameter ändern**.
   Das Dialogfeld "SOAP Request-Einstellungen" wird geöffnet. Sie können nun den Connection Endpoint sowie die SOAP Operation ändern. Bestätigen Sie Ihre Änderungen mit OK.



2. Sie können nun den SOAP Request erneut senden (Menüoption **Soap | SOAP Request an Server senden)**.
   Ein SOAP Response Dokument wird erzeugt und nach einigen Sekunden im Hauptfenster dargestellt.

Bitte beachten Sie:
Ändern Sie die Soap Operation-Einstellungen nur, wenn Ihnen mehrere Aktionen in einer Liste zur Verfügung stehen.

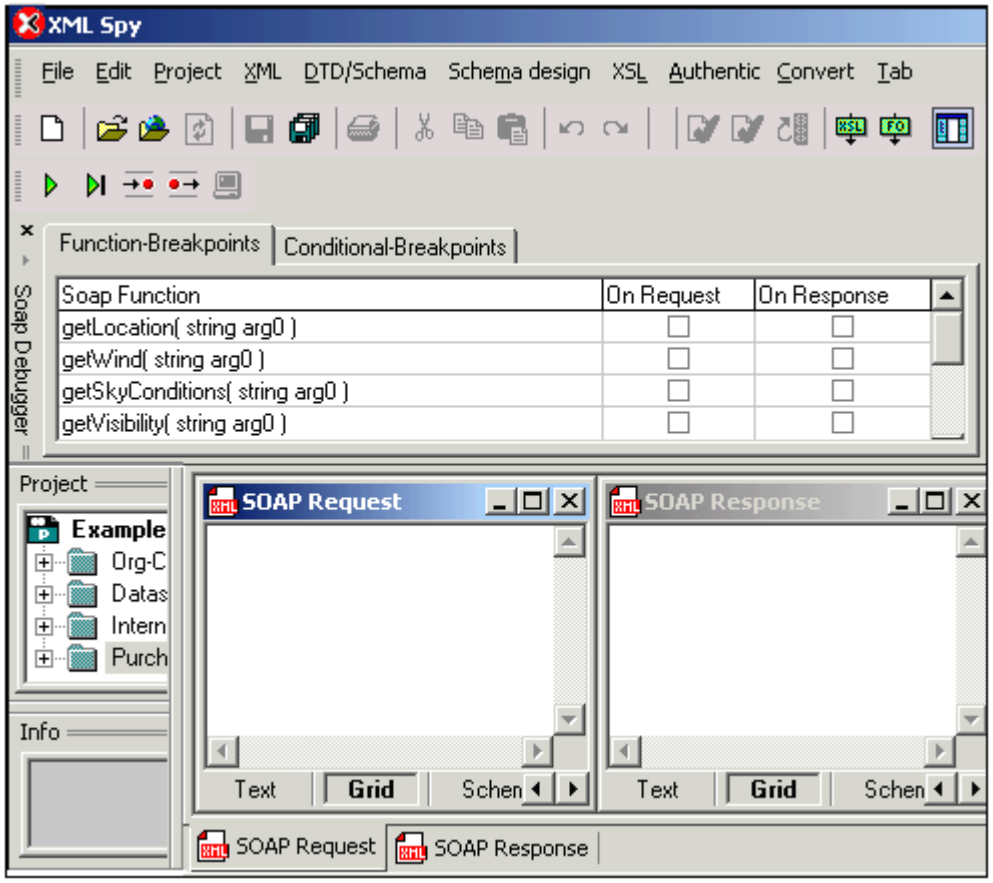
15.14.4 Soap Debugger-Sitzung

Mit diesem Befehl starten Sie eine SOAP Debugger-Sitzung.

- Wenn Sie diesen Befehl ausführen, wird ein Dialogfeld angezeigt. Wählen Sie den Standort der WSDL-Datei (normalerweise eine URL).
- Definieren Sie im nächsten Dialogfeld die Quell- und Ziel-Ports, die für den Debugger Proxyserver und das Webservice benötigt werden.

  Der SOAP Debugger Proxyserver wird geöffnet, ist aber **noch nicht aktiv**. Klicken Sie auf eines der Symbole in der Symbolleiste. Der SOAP Debugger wird gestartet und wartet nun auf Ihre Befehle.

  Weitere Informationen erhalten Sie unter SOAP Debugger .



15.14.5 Starten

Mit diesem Befehl wird der SOAP Proxy Server gestartet. Die Prozedur wird solange in der WSDL-Datei fortgesetzt, bis ein Breakpoint erreicht wird. Das zugehörige SOAP Dokument wird dann in einem der SOAP Dokument-Fenster angezeigt.

15.14.6 Einzelschritt

Mit diesem Befehl können Sie sich schrittweise durch ein- und ausgehende SOAP Requests und Responses bewegen. Bei jedem Request und Response hält der Debugger automatisch an. Der Proxyserver wird auch gestartet, falls dieser inaktiv war.

15.14.7 Beim nächsten Request anhalten

Mit diesem Befehl hält der Debugger beim nächsten SOAP Request an und die Daten werden im SOAP Request Dokument-Fenster dargestellt. Sie können die Daten im Fenster bearbeiten bevor Sie diese an den Webservice senden.

15.14.8 Bei nächstem Response anhalten

Mit diesem Befehl hält der Debugger beim nächsten SOAP Response an und die Daten werden im SOAP Response Dokument Fenster dargestellt. Sie können die Daten im Fenster bearbeiten bevor Sie diese an den Client senden.

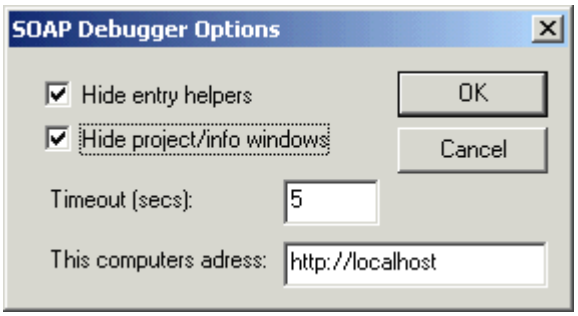
15.14.9 Proxyserver anhalten

Mit diesem Befehl halten Sie den Debugger Proxyserver an.

15.14.10 Soap Debugger Einstellungen

Mit diesem Befehl definieren Sie die Optionen für den SOAP Debugger. Bitte definieren Sie die Optionen **bevor** Sie den Debugger starten.

Ausblenden:
Sie können sowohl die Eingabehilfen als auch das Projektfenster ausblenden, um Ihr SOAP Dokument besser ansehen zu können.



Timeout (sek.)
Die meisten Clients haben eine Timeout-Einstellung für das Einlangen von Daten. Die Standardeinstellung der Zeit, die der Debugger bei dem jeweiligen Breakpoint bleibt, ist bei 5 Sekunden. Eine Datenpaketnachricht (http1.1 100 Continue) wird nach jeder von Ihnen definierten Timeoutperiode an den Client gesendet.

Wenn Sie die Timeoutperiode auf Null setzen, bleiben Sie bei dem Breakpoint, solange Sie wollen.

Adresse des Rechners:
Adresse des Computers, auf dem der SOAP Debugger läuft. Dieser Computer muss für den Client-Computer zugänglich sein.

Geben Sie die Adresse in folgender Form ein: **http://computer Name**, wobei "Computer Name" für die Webadresse oder IP-Adresse des Computers steht, auf dem der SOAP Debugger läuft. Geben Sie in der Adresse keine Port-Nummer an. Der Eintrag "localhost" verweist immer auf Ihren lokalen Rechner.

© 2007 Altova GmbH

15.15 Menü "Extras"

Im Menü Extras können Sie:

- Die Rechtschreibung Ihrer XML-Dokumente überprüfen
- In die Skript-Umgebung von XMLSpy wechseln. Sie können Ihre eigenen Formulare, Makros und Event Handler erstellen, bearbeiten und speichern.
- Derzeit zugewiesene Makros ansehen
- Skripts einem Projekt zuweisen oder zugewiesene Skripts löschen
- zwei beliebige Dateien vergleichen und auf Unterschiede überprüfen
- zwei beliebige Ordner vergleichen und auf Unterschiede überprüfen
- Ihre Version anpassen: eigene Menüleisten, Tastaturkürzel, Menüs und Makros erstellen
- Globale Programm-Einstellungen definieren

15.15.1 Rechtschreibung...

XMLSpy enthält eine integrierte Rechtschreibprüfung.

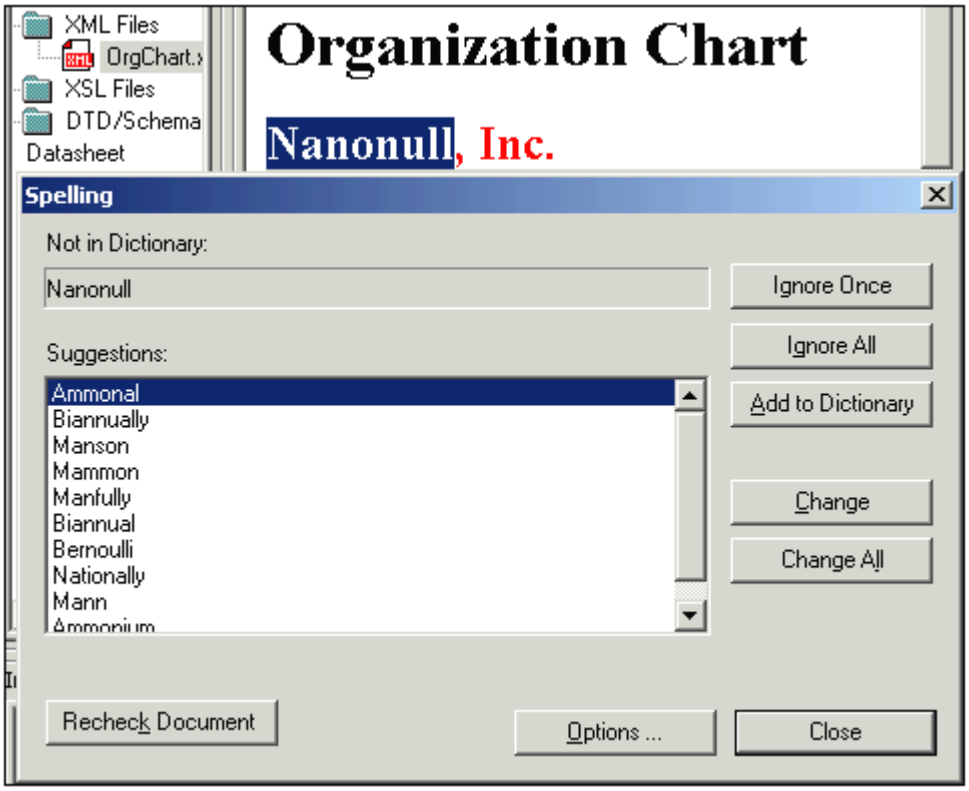
Die Rechtschreibprüfung kann in folgenden Ansichten verwendet werden: Enhanced Grid, Text und Authentic. Bitte beachten Sie, dass jede Art von XML Datei überprüft werden kann. Die Ansicht, in der Sie die Rechtschreibung überprüfen, beeinflusst die Optionen, die für die Rechtschreibprüfung zur Verfügung stehen. Eine Schemadatei (*.xsd) kann in der Text und Enhanced Grid Ansicht, aber nicht in der Schema/WSDL Design-Ansicht überprüft werden.

Die Rechtschreibprüfung in der **Text-** und **Enhanced Grid-Ansicht** bietet Ihnen die meisten Optionen für die Überprüfung. Sie können zusätzlich die Rechtschreibung von folgenden Objekten überprüfen:

- Inhalt von Elementen
- Inhalt von Attributen
- Inhalt von CDaten
- Kommentartexten

  Weiters können Sie definieren, welche Inhalte von Elementen oder Attributen überprüft oder übersprungen werden sollen. Die Einstellungen werden im Dialogfeld " Rechtschreibkontext" vorgenommen.

Mit dem Befehl **Extras | Rechtschreibung...** (Umschalt+F7) öffnen Sie das Dialogfeld für die Rechtschreibprüfung. Das aktive XML-Dokument wird automatisch auf Rechtschreibfehler geprüft (in diesem Screenshot das OrgChart.XML-Dokument in der Authentic-Ansicht).



Nicht im Wörterbuch
Dieses Textfeld enthält ein Wort aus dem Text, welches in keinem der existierenden Wörterbücher (Standardwörterbuch und benutzerdefinierte Wörterbücher) vorkommt. Sie können den Text hier bearbeiten und auf **Ändern** klicken, um ihn im XML-Dokument zu ändern. Das Wort wird im XML-Dokument zur besseren Übersichtlichkeit markiert anzeigt. Die Schaltflächen werden nun aktiviert, und Sie können entscheiden, wie Sie fortfahren möchten.

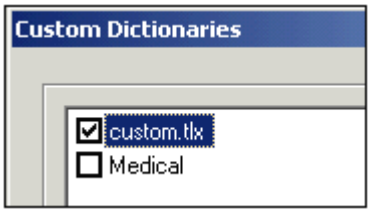
Vorschläge
Dieses Listenfeld zeigt eine Reihe von Wörtern an, die dem unbekannten Wort am nähesten kommen (aus allen verfügbaren Wörterbüchern). Durch Doppelklicken auf einen Eintrag wird dieser anstelle des unbekannten Wortes eingefügt und die Rechtschreibprüfung wird fortgesetzt.

Einmal ignorieren
Mit diesem Befehl können Sie das unbekannte Wort bei seinem ersten Auftreten ignorieren und die Rechtschreibprüfung fortsetzen. Sollte das Wort ein weiteres Mal im Dokument vorkommen, wird die Rechtschreibprüfung es dann wieder als unbekannt markieren.

Immer ignorieren
Mit diesem Befehl wird das Wort durch das ganze Dokument ignoriert.

Ins Wörterbuch
Mit diesem Befehl können Sie unbekannte Wörter in das aktive **benutzerdefinierte Wörterbuch** einfügen. Das aktive benutzerdefinierte Wörterbuch ist das Wörterbuch, das im Dialogfeld "Benutzerdefinierte Wörterbücher" mit einem Häkchen versehen ist. Zum Öffnen dieses Dialogfelds klicken Sie auf **Optionen,** anschließend **Rechtschreiboptionen** und **Benutzerdefinierte Wörterbücher**.

© 2007 Altova GmbH



Ändern
Mit diesem Befehl ersetzen Sie das im XML-Dokument markierte Wort durch den markierten Eintrag aus der Auswahlliste der Vorschläge.

Alle ändern
Mit diesem Befehl ersetzen Sie das Wort an allen Stellen im XML-Dokument, an denen es vorkommt.

Dokument erneut kontrollieren
Das Dokument wird vom Anfang an neu überprüft.

Schließen
Mit diesem Befehl beenden Sie die Rechtschreibprüfung.

Einstellungen...
Mit diesem Befehl öffnen Sie ein Dialogfeld in der aktuellen Ansicht:

- In der Text oder Enhanced Grid-Ansicht und in der Authentic-Ansicht wird das Dialogfeld Rechtschreibkontext geöffnet.

15.15.2 Rechtschreibung - Optionen...

Mit diesem Befehl öffnen Sie das Dialogfeld **Optionen** oder **Rechtschreibkontext**, je nachdem, in welcher Ansicht Sie Ihr XML-Dokument derzeit bearbeiten.

Dialogfeld "Optionen" (Browser-Ansicht und Schema/WSDL-Ansicht)
Bei der Ansicht eines XML-Dokuments in der Browser-Ansicht oder der Schema/WSDL-Ansicht wird mit diesem Befehl das Dialogfeld "Optionen" geöffnet. Über dieses Dialogfeld können Sie Optionen für die Rechtschreibprüfung zu definieren.

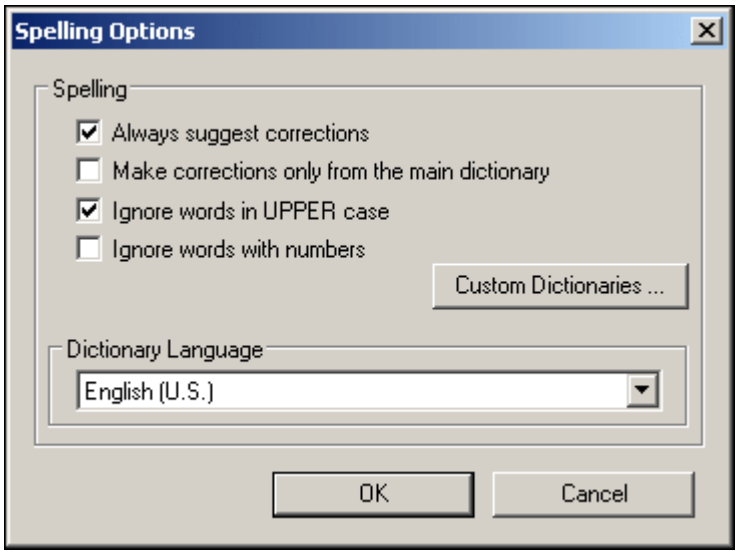
Korrektur immer vorschlagen:
Wenn Sie dieses Kontrollkästchen aktivieren, werden alle Vorschläge (aus allen Wörterbüchern) in der Vorschlagsliste angezeigt. Wenn Sie diese Option deaktivieren, werden keinen Vorschläge angezeigt.

Korrekturen nur aus dem Hauptwörterbuch übernehmen:
Bei Aktivierung dieser Option werden nur Vorschläge angezeigt, die im Hauptwörterbuch zur Verfügung stehen. Weiters wird die Schaltfläche "Benutzerdefinierte Wörterbücher" deaktiviert.

Großschreibung ignorieren:
Großgeschriebene Wörter werden in der Rechtschreibprüfung ignoriert.

Wörter mit Ziffern ignorieren:
Wörter, die Ziffern enthalten, werden in der Rechtschreibprüfung ignoriert.

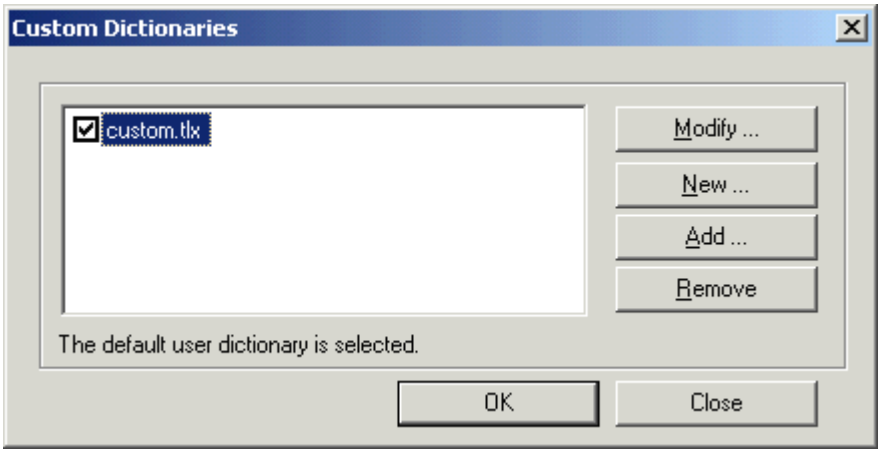
Sprache Wörterbuch
Wählen Sie eine Sprache im Dropdown-Listenfeld, in der die Rechtschreibprüfung erfolgen soll. Die standardmäßig ausgewählte Sprache ist derzeit Englisch. Weitere Sprachen werden demnächst zum Download von Altovas Webseite zur Verfügung stehen.



Benutzerdefinierte Wörterbücher
Mit dieser Schaltfläche können Sie:

- existierende Wörterbücher bearbeiten (Einträge hinzufügen oder löschen)
- neue Wörterbücher erstellen
- ein existierendes Wörterbuch hinzufügen
- ein existierendes Wörterbuch löschen

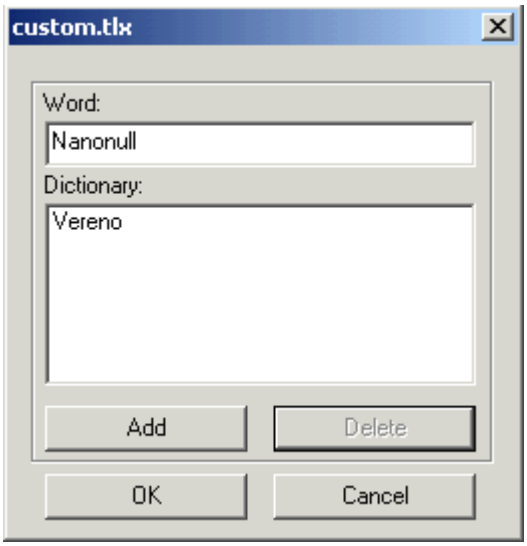
Wenn Sie die Rechtschreibprüfung starten, werden alle im Listenfeld angezeigten benutzerdefinierten Wörterbücher durchsucht. Mit dem Befehl **Entfernen** können Sie bestimmte Wörterbücher deaktivieren. Diese werden während der Rechtschreibprüfung nicht verwendet.



So fügen Sie einen neuen Wörterbucheintrag hinzu:

1. Klicken Sie im Dialogfeld "Benutzerdefinierte Wörterbücher" auf den Namen des benutzerdefinierten Wörterbuches, zu dem Sie einen Eintrag hinzufügen möchten, und klicken Sie auf **Ändern...**
   Das ausgewählte Wörterbuch wird geöffnet (in unserem Beispiel, custom.tlx). Falls Sie kein Wörterbuch ausgewählt haben, werden Sie dazu aufgefordert.

© 2007 Altova GmbH



2. Klicken Sie auf das Feld "Wort:" und geben Sie den neuen Wörterbucheintrag ein (die Schaltfläche "Hinzufügen" wird hierdurch automatisch aktiviert).
3. Klicken Sie auf "Hinzufügen", um den Eintrag in das Wörterbuch einzufügen.
4. Klicken Sie auf OK.

So löschen Sie einen Eintrag aus dem Wörterbuch:

1. Klicken Sie im Dialogfeld "Benutzerdefinierte Wörterbücher" auf den Namen des Wörterbuches aus dem Sie einen Eintrag löschen möchten und klicken Sie auf **Ändern..**. Daraufhin wird das im Listenfeld markierte Wörterbuch geöffnet (in diesem Fall custom.tlx). Wenn kein Wörterbuch ausgewählt wurde, werden Sie aufgefordert eines auszuwählen.
2. Klicken Sie auf das zu löschende Wort im Listenfeld, und klicken Sie auf **Löschen**.
3. Klicken Sie auf **OK**.

So fügen Sie ein neues Wörterbuch ein:

1. Klicken Sie im Dialogfeld "Benutzerdefinierte Wörterbücher" auf **Neu**, und geben Sie den Namen des neuen benutzerdefinierten Wörterbuches im Feld "**Dateiname...**" ein.
2. Klicken Sie auf **Speichern** um das Wörterbuch zu speichern.
3. Über die Schaltfläche "Hinzufügen" oder "Ins Wörterbuch" können Sie nun während der Rechtschreibprüfung neue Einträge in das neue Wörterbuch einfügen.

So fügen Sie ein existierendes Wörterbuch ein:

Verwenden Sie diese Option um zuvor deaktivierte Wörterbücher (oder Wörterbücher von anderen Anbietern) wieder einzufügen.

1. Klicken Sie im Dialogfeld "Benutzerdefinierte Wörterbücher" auf "Hinzufügen" und geben Sie den Namen des Wörterbuchs in das Feld **Dateiname** ein. Wörterbücher erkennen Sie an der Erweiterung *.**tlx**.
2. Klicken Sie auf "Öffnen", um das Wörterbuch zu aktivieren.

Bitte beachten Sie:

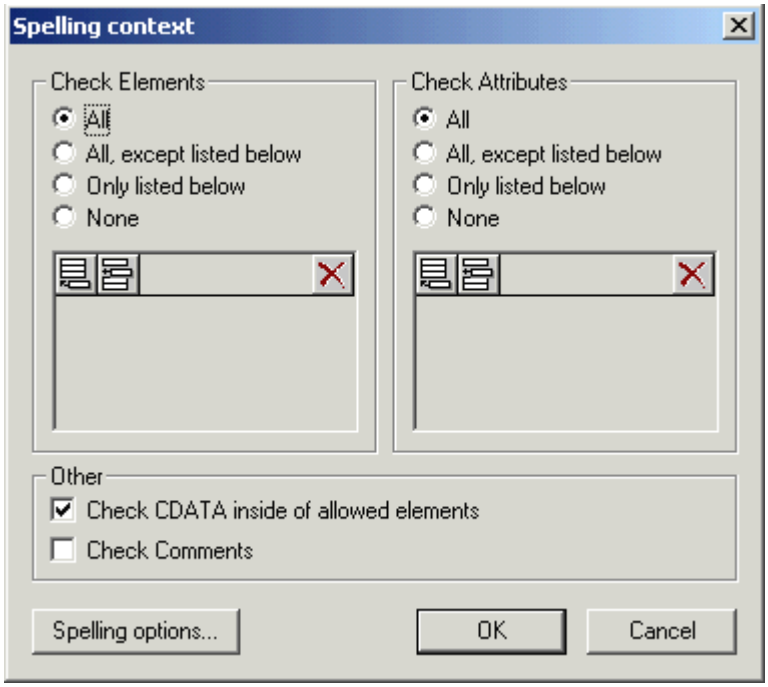
Wörterbücher müssen nicht unbedingt eine *.tlx Erweiterung haben. Das Wörterbuch kann eine beliebige oder auch keine Erweiterung haben und trotzdem in die Liste der Wörterbücher aufgenommen werden.

So entfernen Sie ein Wörterbuch:

- Klicken Sie im Dialogfeld "Benutzerdefinierte Wörterbücher" auf den Namen des Wörterbuchs und dann die Schaltfläche "Entfernen". Das Wörterbuch wird aus der Liste, aber nicht von der Festplatte gelöscht.

Dialogfeld Rechtschreibkontext - **Text-**und **Enhanced Grid-Ansicht**
Wenn Sie Ihr XML-Dokument in der Text- oder Enhanced Grid-Ansicht bearbeiten, können Sie mit diesem Befehl das Dialogfeld "Rechtschreibkontext" öffnen. Wenn Sie die Rechtschreibung in der **Text-** und **Enhanced Grid-Ansicht** starten, haben Sie die meisten Optionen zur Verfügung. So können Sie zum Beispiel folgende Inhalte auf Rechtschreibung überprüfen:

- Inhalte von Elementen
- Inhalte von Attributen
- Inhalte von CDaten
- Kommentartexte

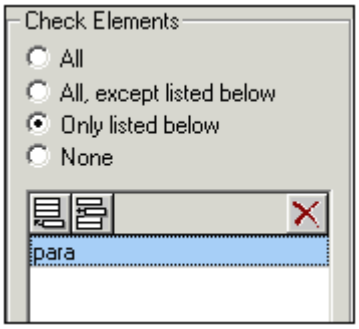


Die Schaltfläche Rechtschreiboptionen... öffnet das Dialogfeld Einstellungen, wo Sie Einstellungen für die Rechschreibprüfung definieren können.

So definieren Sie, welcher Inhalt von Attributen oder Elementen überprüft werden soll:

1. Klicken Sie auf das Optionsfeld "Alle außer unten aufgelistete" oder "Nur unten aufgelistete".
2. Klicken Sie auf das Symbol "Element/Attribut anhängen" , und geben sie den Tag-Namen ein, dessen Inhalt überprüft werden soll (z.B. para). Bestätigen Sie mit der Eingabetaste.

© 2007 Altova GmbH



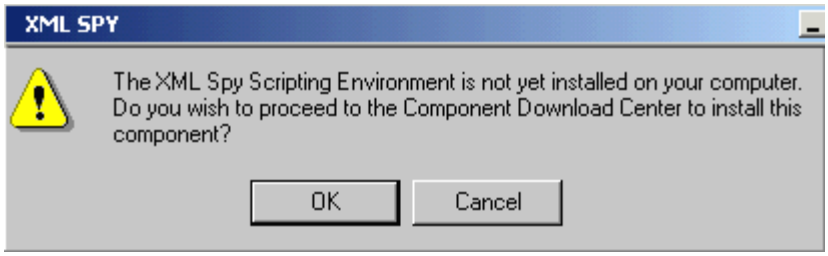
3. Klicken Sie auf die OK Schaltfläche um Ihre Einstellungen zu bestätigen, und bestätigen Sie diese nochmals im Dialogfeld "Rechtschreibung" um die Rechtschreibprüfung mit den neuen Einstellungen zu starten.
   In diesem Fall wird **nur der Text innerhalb der Para-Tags**, sowie dessen Attribute auf Rechtschreibfehler überprüft.

So löschen Sie einen Tag-Namen:

- Wählen Sie den Tag-Namen, den Sie löschen möchten, und drücken Sie die Entfernen- oder die Rücktaste Ihrer Tastatur.

15.15.3 Skripting-Umgebung

Die Skripting-Umgebung ist derzeit eine downloadbare Komponente. Wenn Sie die Komponente noch nicht heruntergeladen und installiert haben, werden Sie dazu aufgefordert, sobald Sie diesen Menübefehl wählen.



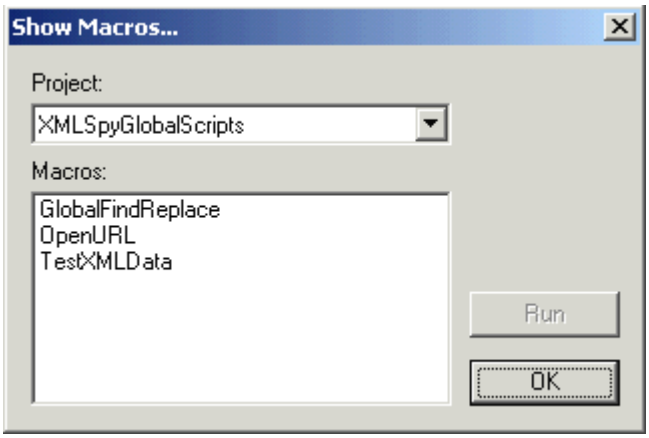
Klicken Sie auf OK um mit dem Download-Center auf Altovas Webseite zu verbunden zu werden. Hier wählen Sie dann die Komponente Skript-Umgebung.

Nachdem Sie die Komponente heruntergeladen haben, doppelklicken Sie auf **Spyscripting.exe** um diese zu installieren.

Die Scripting-Umgebung, ist ein vordefiniertes globales Skript-Projekt, und wird beim ersten Starten des Formular-Editors aktiviert. Weitere Informationen zu diesem Thema finden Sie im Skript-Abschnitt der vorliegenden Dokumentation.

15.15.4 Makros...

Mit diesem Befehl wird das Dialogfeld mit der Liste aller Makros angezeigt, die mit dem globalen Skriptprojekt und mit dem aktuellen Projekt in XMLSpy assoziiert sind.



Die Schaltfläche "Starten" im Dialogfeld ruft das markierte Makro auf.

15.15.5 Projekt

Mit diesem Befehl können Sie Skripts einem bestimmten Projekt zuweisen (oder löschen).

Bitte beachten Sie:
Diese Menüoptionen sind nur aufrufbar, wenn Sie ein Projekt in XMLSpy geöffnet haben.

Skripts zuweisen

Mit diesem Befehl können Sie einem Skiptprojekt ein Skript zuweisen.

Projekt-Skripts entkoppeln

Mit diesem Befehl können Sie in einem Skript-Projekt eine Skirpt-Zuweisung löschen.

Projekt-Skripts aktivieren

Mit diesem Befehl können Sie zeitweilig ein Projekt-Skript aktivieren oder deaktivieren.

15.15.6 Vergleiche

XMLSpy bietet eine Vergleichs- (oder Differencing)-Funktionalität, mit der XML- und Textdateien sowie Ordner auf Unterschiede überprüft werden können.

Das Menü Extras enthält die folgenden Menüpunkte, mit Hilfe derer Sie Vergleiche an Dateien und Ordnern durchführen können:

- Geöffnete Datei vergleichen mit
- Verzeichnisse vergleichen
- Vergleichseinstellungen

Diese Befehle werden in den folgenden Unterabschnitten näher beschrieben.

Geöffnete Datei vergleichen mit...

Mit diesem Befehl können Sie eine geöffnete Datei mit einer anderen Datei vergleichen. Beim Vergleich werden beide Dateien im Hauptfenster senkrecht nebeneinander angezeigt, wobei die Unterschiede in den beiden Dateien grün markiert sind. Dateien können als XML-Dokumente

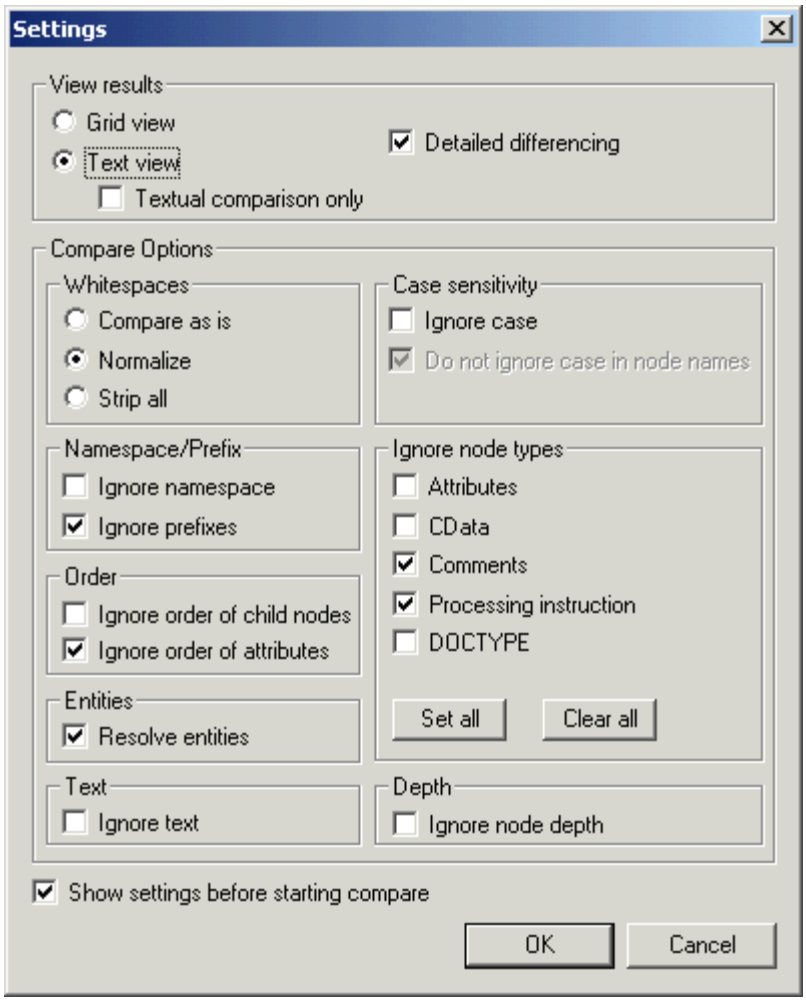
 © 2007 Altova GmbH

(unter Berücksichtigung der Struktur und Semantik der Tags) oder als Textdokumente verglichen werden.

So vergleichen Sie eine geöffnete Datei mit einer anderen Datei:

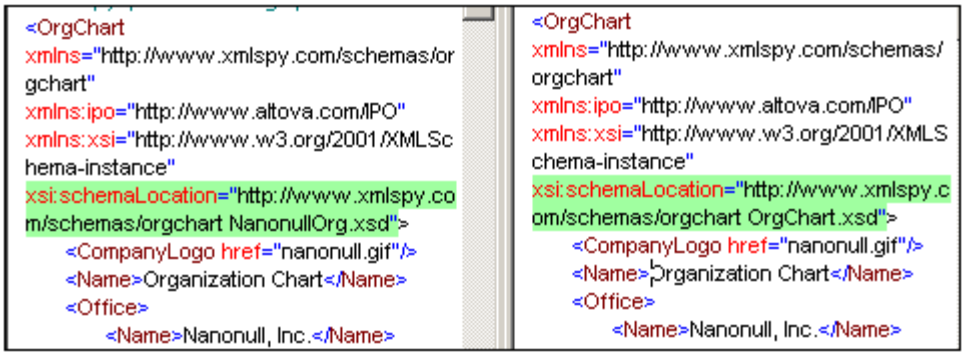
1. Wenn die Datei im Hauptfenster geöffnet ist (im Hauptfenster kann immer nur jeweils eine Datei aktiv sein), klicken Sie auf **Extras | Geöffnete Datei vergleichen mit...**. Daraufhin wird das folgende Dialogfeld angezeigt:

2. Suchen Sie die Datei, mit der Sie die geöffnete Datei vergleichen möchten, und wählen Sie sie aus. Wenn die gewünschte Datei im Hauptfenster bereits geöffnet (aber nicht aktiv) ist oder wenn sich die Datei in einem Projekt befindet, klicken Sie auf die Schaltfläche **Fenster** und wählen Sie die Datei aus.
3. Bestätigen Sie mit OK. Die zweite Datei wird geöffnet und das Dialogfeld "Einstellungen" wird angezeigt:

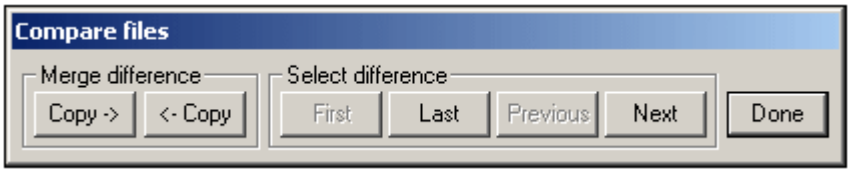


Diese Einstellungen werden unter Vergleichseinstellungen beschrieben. Wenn Sie nicht möchten, dass dieses Dialogfeld jedes Mal angezeigt wird, wenn Sie einen Dateivergleich starten, deaktivieren Sie das Kontrollkästchen "Einstellungen vor Vergleich anzeigen" am unteren Rand des Dialogfelds.

4. Wählen Sie die gewünschten Einstellungen aus und klicken Sie auf OK. Die beiden Dateien werden nun im Hauptfenster angezeigt. Zeilen, die Unterschiede aufweisen, erscheinen grün markiert. Der jeweils markierte Unterschied wird in beiden Dateien in einem dunkleren Grün angezeigt.



Außerdem wird das Steuerungsfenster "Dateien vergleichen" angezeigt:



Um den nächsten Unterschied auszuwählen, klicken Sie auf die Schaltfläche "**Weiter**". Mit Hilfe der Schaltflächen **Erste**, **Letzte** und **Vorherig** können Sie von einem Unterschied zu anderen navigieren und einen Unterschied markieren.
Im Bereich "Unterschiede zusammenführen" können Sie die markierten unterschiedlichen Stellen von einer Datei in die andere kopieren. Um das Zusammenführen von Dateien zu aktivieren, müssen die folgenden Vergleichoptionen eingestellt sein: Die Option "Detaillierter Vergleich" muss aktiviert sein und die Option "Node-Tiefe ignorieren" muss deaktiviert sein. Wenn Sie einen Zusammenführvorgang rückgängig machen wollen, stoppen Sie den Vergleich, wählen Sie die Datei, in der die Änderung rückgängig gemacht werden soll, und klicken Sie auf den Befehl **Bearbeiten | Rückgängig** oder drücken Sie **Strg + Z**.

5. Nach Abschluss der Überprüfung und/oder des Zusammenführens von Unterschieden, klicken Sie auf "Fertig".

Bitte beachten Sie:

- Während des Vergleichsvorgangs können keine Bearbeitungen vorgenommen werden und auch die Ansicht kann nicht gewechselt werden. Bei Versuchen, die Ansicht einer der Dateien zu wechseln oder eine Datei zu bearbeiten, wird eine Meldung angezeigt mit der Warnung, dass dadurch der Vergleich beendet wird.
- Sie können während eines Vergleichs die Vergleichseinstellungen ändern (durch Auswahl von **Extras | Vergleichseinstellungen...**), doch werden die Änderungen erst beim nächsten Vergleichsvorgang wirksam; die neuen Einstellungen haben keinen Einfluss auf die aktuelle Sitzung.

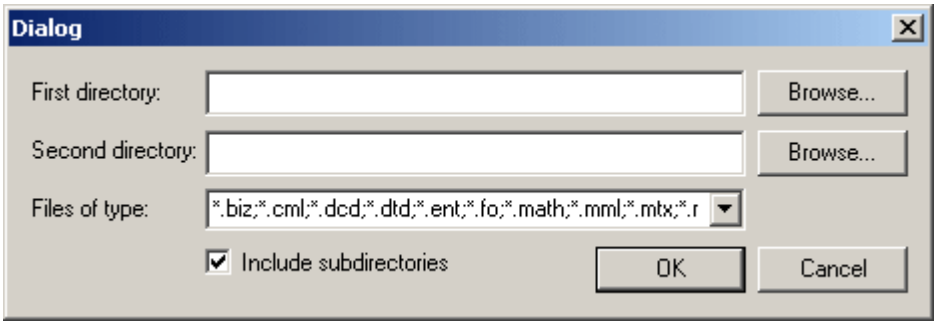
© 2007 Altova GmbH

Verzeichnisse vergleichen...

Mit dem Befehl "Verzeichnisse vergleichen..." können Sie zwei Verzeichnisse einschließlich oder ausschließlich ihrer Unterverzeichnisse vergleichen. Der Vergleich von Verzeichnissen dient dazu, festzustellen, welche Dateien fehlen und ob Dateien desselben Namens Unterschiede aufweisen oder nicht.

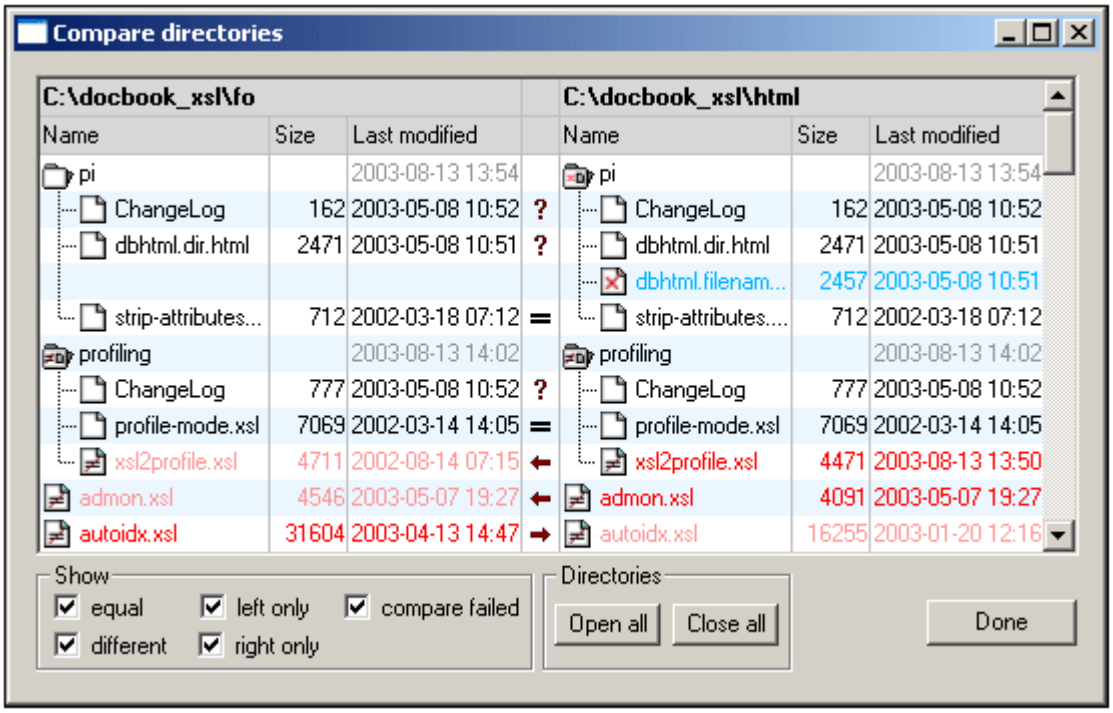
So vergleichen Sie zwei Verzeichnisse:

1. Klicken Sie auf **Extras | Verzeichnisse vergleichen**.Daraufhin wird das folgende Dialogfeld angezeigt.



2. Suchen Sie die zu vergleichenden Verzeichnisse und aktivieren Sie das Kontrollkästchen **Unterverzeichnisse berücksichtigen**, wenn Unterverzeichnisse ebenfalls einbezogen werden sollen.
3. Wählen Sie die zu vergleichenden Dateitypen aus.
4. Klicken Sie auf OK. Das Dialogfeld "Einstellungen" wird angezeigt (Beschreibung siehe Vergleichseinstellungen).
5. Wählen Sie die gewünschten Einstellungen für den Dateivergleich aus.
6. Klicken Sie auf OK. Daraufhin erscheint ein Dialogfeld, in dem Sie den Verlauf des Vergleichs beobachten können.

Das Ergebnis des Vergleichs wird in einem Fenster angezeigt und sieht etwa folgendermaßen aus:



Verzeichnissymbole
Alle Verzeichnisnamen werden schwarz angezeigt.

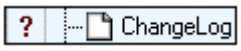
Dieses Verzeichnis enthält Dateien, die entweder alle nicht verglichen werden konnten oder sich nicht von der gleichnamigen Datei im anderen Verzeichnis unterscheiden.

Dieses Verzeichnis enthält eine oder mehrere Dateien, die im Verzeichnis, mit dem es verglichen wird, nicht vorhanden sind.

Dieses Verzeichnis enthält eine oder mehrere Dateien, die sich von der gleichnamigen Datei im anderen Verzeichnis unterscheiden.

Bitte beachten Sie: Wenn ein Verzeichnis sowohl eine Datei enthält, die im verglichenen Verzeichnis nicht existiert, als auch eine Datei, die sich von der gleichnamigen Datei im verglichenen Verzeichnis unterscheidet, wird für das Verzeichnis das Symbol angezeigt.

Dateisymbole
Je nach Vergleichsstatus werden die Dateinamen in unterschiedlichen Farben angezeigt. Zur Kennzeichnung werden die folgenden Farben verwendet:



Diese Datei kann nicht (mit der gleichnamigen Datei im anderen Verzeichnis) verglichen werden. In der mittleren Spalte erscheint ein Fragezeichen. Der Dateiname wird schwarz angezeigt.

Diese Datei unterscheidet sich nicht von der gleichnamigen Datei im anderen Verzeichnis. In der mittleren Spalte wird ein Ist gleich-Zeichen angezeigt. Der Dateiname erscheint schwarz.

Diese Datei existiert im verglichenen Verzeichnis nicht. Die mittlere Spalte ist leer. Der Dateiname wird blau angezeigt.

Diese Datei weist Unterschiede zur gleichnamigen Datei im

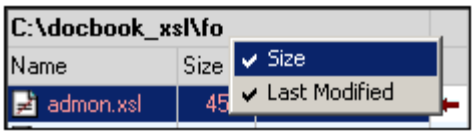
 © 2007 Altova GmbH

verglichenen Verzeichnis auf und die letzte Änderung dieser Datei ist jüngeren Datums als die letzte Änderung der verglichenen Datei. In der mittleren Spalte ist ein Pfeil zu sehen, der auf die jüngere Datei weist. Der Dateiname erscheint dunkelrot.

Diese Datei weist Unterschiede zur gleichnamigen Datei im verglichenen Verzeichnis auf und die letzte Änderung ist älteren Datums als die letzte Änderung der verglichenen Datei. In der mittleren Spalte wird ein Pfeil angezeigt, der in Richtung dieser (jüngeren) Datei weist. Der Dateiname erscheint hellrot.

Anzeigeoptionen

- Wählen Sie aus, welche Dateien angezeigt werden sollen. Aktivieren bzw. deaktivieren Sie dazu im Anzeigebereich am unteren Rand des Ergebnisfensters die entsprechenden Optionen.
- Öffnen oder schließen Sie alle Unterverzeichnisse durch Klicken auf die entsprechende Schaltfläche im Bereich "Verzeichnisse".
- Erweitern bzw. reduzieren Sie Unterverzeichnisse durch Doppelklicken auf das entsprechende Ordnersymbol.
- Die Spalten "Größe" und "Zuletzt geändert" können ein und ausgeblendet werden, indem Sie mit der rechten Maustaste auf eine Dateikopfzeile klicken und auf "Größe" und "Zuletzt geändert" klicken.

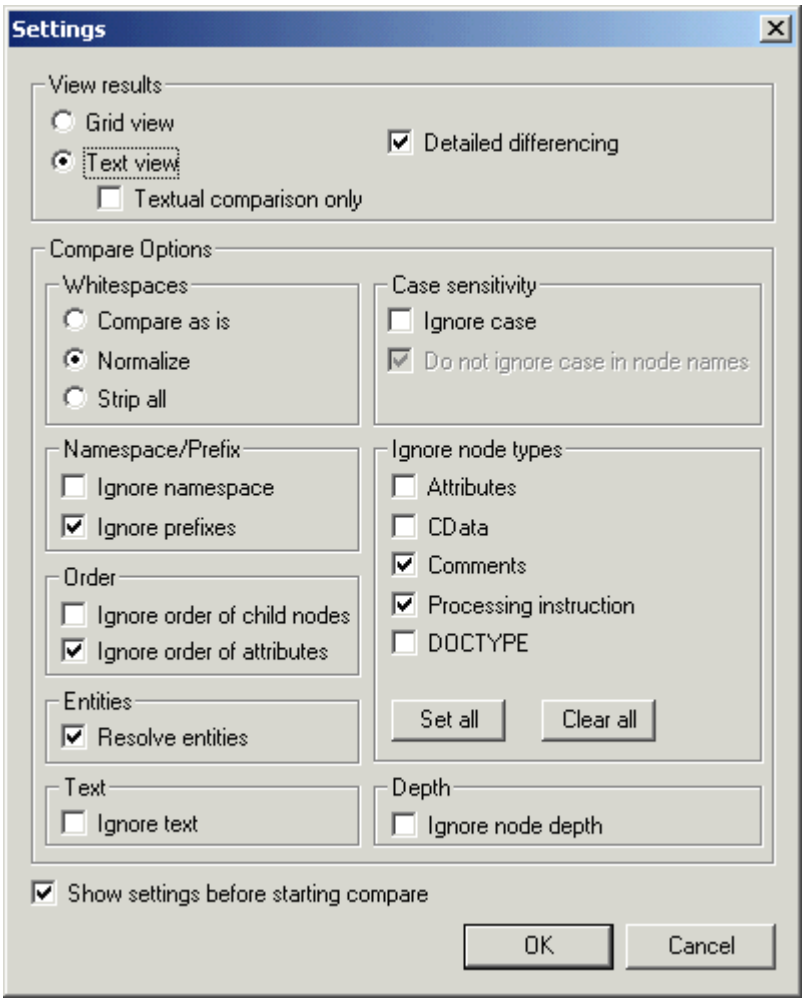


- Durch Ziehen der Spalten können Sie die Spaltenbreite ändern.
- Sie können das Ergebnisfenster maximieren, minimieren oder seine Größe anpassen.

Vergleichen und Zusammenführen von Dateien
Wenn Sie auf eine Zeile doppelklicken, werden beide in dieser Zeile befindlichen Dateien im Hauptfenster geöffnet, und an den beiden Dateien wird ein Dateivergleich durchgeführt. Sie können anschließend genau wie in einer normalen Vergleichssitzung fortfahren (siehe "Geöffnete Datei vergleichen mit").

Vergleichsoptionen

Wenn Sie auf **Vergleichsoptionen** klicken, wird das unten gezeigte Dialogfeld "Einstellungen" angezeigt, in dem Sie die Einstellungen für Ihre Vergleichssitzungen festlegen. Die Einstellungen, die beim Start einer Vergleichssitzung aktiv sind, sind diejenigen, die auf diese Sitzung angewendet werden.



Ergebnisse anzeigen
Wählt die Ansicht aus, in der die Ergebnisse angezeigt werden sollen. Sie haben die Wahl zwischen folgenden Möglichkeiten:

- Grid-Ansicht (XML-Vergleich)
- Textansicht; die Option "Nur Textvergleich" ist deaktiviert (XML-Vergleich)
- Textansicht; die Option "Nur Textvergleich" ist aktiviert (Textvergleich)

Wenn Sie eine Ansicht auswählen, die den XML-Vergleich unterstützt, werden die Dokumente als XML-Dokumente behandelt und die XML-Vergleichsoptionen werden aktiviert. Wird Textvergleich ausgewählt, werden nur diejenigen Optionen aktiviert, die den Textvergleich betreffen (Whitespaces und Groß- und Kleinschreibung); alle anderen Vergleichsoptionen sind deaktiviert.

Bitte beachten Sie: Sie können Dateien sowohl in der Grid-Ansicht als auch in der Textansicht und sowohl im XML- als auch im Textvergleichsmodus zusammenführen. Wenn Sie eine Zusammenführung rückgängig machen möchten, stoppen sie den Vergleich, wählen Sie die Datei aus, in der die Änderung rückgängig gemacht werden soll und klicken Sie auf **Bearbeiten | Rückgängig** oder drücken Sie **Strg + Z**.

Detaillierter Vergleich
Ist diese Option deaktiviert, werden Unterschiede zwischen unmittelbaren Geschwisterelementen als ein einziger Unterschied dargestellt und die Zusammenführung wird

 © 2007 Altova GmbH

deaktiviert. Ist die Option aktiv, werden Unterschiede zwischen unmittelbaren Geschwisterelementen als separate Unterschiede markiert und die Zusammenführung wird aktiviert.

Bitte beachten Sie: Um die Zusammenführung von Dateien zu aktivieren, muss die Option "Detaillierter Vergleich" aktiviert sein.

Whitespaces
Bei Leerzeichen handelt es sich um Leerzeichen, Tabulatorzeichen, Wagenrücklauf und Zeilenvorschub. Die Optionen in diesem Bereich dienen zum Vergleichen von Dateien, bei denen die Leerzeichen identisch sind, zum Vergleich von Dateien bei denen die Leerzeichen vereinheitlicht wurden (d.h. alle aufeinander folgenden Leerzeichen werden als ein Leerzeichen betrachtet) und zum Vergleich von Dateien, bei denen alle Leerzeichen entfernt wurden. Die Option "Whitespaces" steht sowohl für XML- als auch für Textvergleiche zur Verfügung.

Groß-/Kleinschreibung
Ist das Kontrollkästchen "Groß-/Kleinschreibung ignorieren" aktiviert, haben Sie die Option, die Groß- und Kleinschreibung in Node-Namen zu ignorieren oder zu berücksichtigen (nur bei XML-Vergleichen). Die Option "Groß-/Kleinschreibung" steht sowohl für XML- als auch für Textvergleiche zur Verfügung.

Namespace/Präfix
Diese Optionen dienen zum Ignorieren von Namespaces und Präfixen beim Suchen nach Unterschieden.

Reihenfolge
Wenn die Option "Child-Node Reihenfolge ignorieren" ausgewählt ist, spielt die Position der Child Nodes relativ zueinander keine Rolle. Der Vergleich wird an der Gesamtheit aller Child Nodes durchgeführt und wenn der einzige Unterschied zwischen einem Child Node in dem einen Dokument und einem Child Node in dem damit verglichenen Dokument die relative Position im Nodeset ist, so wird dieser Unterschied ignoriert. Beachten Sie jedoch, dass jeder Child-Element-Node an seinem Namen, seinen Attributen und seiner Position identifiziert wird. Haben mehrere Nodes im Geschwister-Nodeset denselben Namen, so werden diese Nodes anhand ihrer Position identifiziert, selbst wenn die Option "Child-Node Reihenfolge ignorieren" ausgewählt ist. Diese Option wird auf jede Ebene separat angewendet.

Ist die Option "Child-Node Reihenfolge ignorieren" deaktiviert, so werden die Unterschiede in der Reihenfolge der Nodes als Unterschiede markiert.

Des Weiteren steht eine Option zum Ignorieren der Reihenfolge der Attribute zur Verfügung. Sie wird auf die Reihenfolge der Attribute eines einzelnen Elements angewendet.

Entities
Ist die Option "Entities auflösen" ausgewählt, werden alle Entitäten im Dokument aufgelöst. Andernfalls werden die Dateien mit den Entitäten in ihrem gegenwärtigen Zustand verglichen.

Text
Wenn Sie die Option "Text ignorieren" auswählen, werden die Unterschiede in einander entsprechenden Text-Nodes nicht berücksichtigt.

Node-Typen ignorieren
Aktivieren Sie das Kontrollkästchen für die Node-Typen, die beim Dateivergleich nicht berücksichtigt werden sollen. Zur Wahl stehen die Node-Typen Attribute, CDATA, Comments, Processing Instructions und DOCTYPE statements.

Tiefe
Ist die Option "Node-Tiefe ignorieren" ausgewählt, so wird die Tiefe eines Elements (d.h.

weitere Ebenen mit Nachfahren dieses Elements) relativ zur Tiefe des entsprechenden Elements in der verglichenen Datei ignoriert. Um die Zusammenführung zu ermöglichen, muss diese Option deaktiviert werden.

Einstellungen vor Vergleich anzeigen
Wenn Sie diese Option aktivieren, wird das Dialogfeld "Einstellungen" jedes Mal angezeigt, wenn Sie auf den Befehl "**Geöffnete Datei vergleichen mit"** klicken. Auf diese Art können Sie die Einstellungen vor jedem Vergleich überprüfen und ändern.

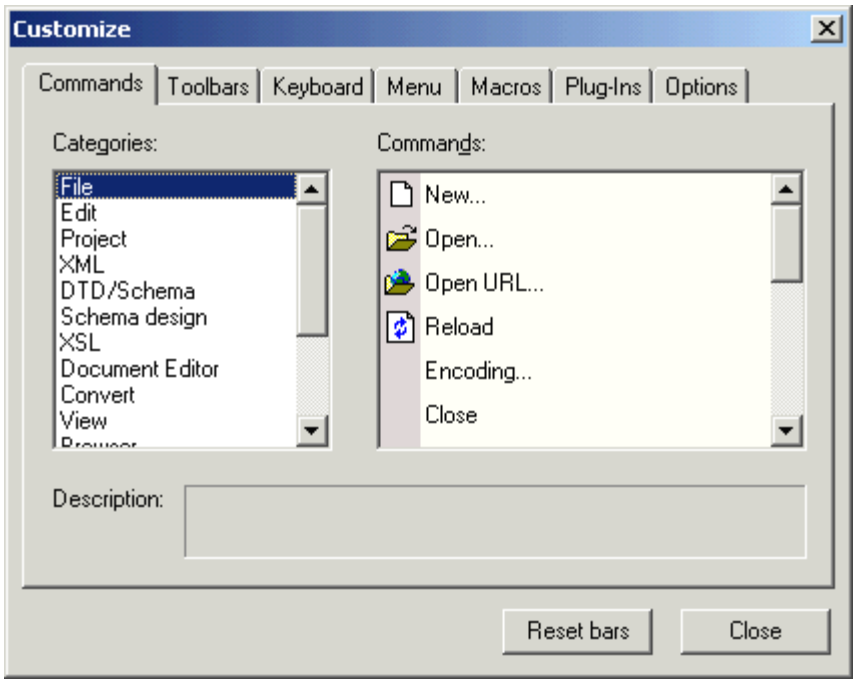
Ist dieses Kontrollkästchen deaktiviert, wird der Vergleich direkt gestartet, wenn Sie den entsprechenden Befehl wählen.

15.15.7 Anpassen...

Mit diesem Befehl können Sie XMLSpy an Ihre eigenen Bedürfnisse anpassen.

Befehle

Auf dem Register Befehle können Sie Ihre Menüs oder Symbolleisten anpassen.



So fügen Sie einen Befehl zu einer Symbolleiste oder einem Menü hinzu:

1. Öffnen Sie das Dialogfeld über die Menübefehle **Extras | Anpassen.** Daraufhin wird das Dialogfeld "Anpassen" angezeigt.
2. Wählen Sie die Befehlskategorie **Alle Befehle** im Kategorie-Listenfeld. Die verfügbaren Befehle werden im entsprechenden Listenfeld angezeigt.
3. Markieren Sie einen Befehl aus dem Listenfeld, und ziehen Sie ihn in ein existierendes Menü oder eine Symbolleiste.
4. Ein I-Zeichen wird angezeigt, wenn Sie den Cursor an eine Stelle bewegen, an die der ausgewählte Befehl gestellt werden kann.
5. Lassen Sie die Maustaste los, um den Befehl an der gewünschten Stelle einzufügen.

 © 2007 Altova GmbH

- An der Spitze des Mauszeigers wird beim Ziehen eines Befehls eine kleine Schaltfläche angezeigt. Das Zeichen unterhalb des Zeigers zeigt an, dass der Befehl nicht an diese Position gestellt werden kann.
- Das Zeichen verschwindet, sobald Sie eine Position erreichen, an die der Befehl gestellt werden kann (innerhalb eines Menüs oder einer Symbolleiste).
- Wenn Sie den Mauszeiger über ein Menü halten, wird dieses geöffnet, so dass Sie den Befehl an einer beliebigen Stelle im Menü einfügen können.
- Befehle können in Menüs oder Symbolleisten eingefügt werden. Wenn Sie eine eigene Symbolleiste erzeugt haben, können Sie nun die gewünschten Befehle einfügen.

Bitte beachten Sie:
Sie können mit dieser Methode auch die Befehle der **Kontextmenüs** bearbeiten (rechtsklicken Sie auf eine beliebige Stelle im Fenster, und das Kontextmenü wird geöffnet). Klicken Sie auf das Register Menü und wählen Sie aus dem Dropdown-Listenfeld das gewünschte Kontextmenü aus.

So löschen Sie einen Befehl oder ein Menü:

1. Öffnen Sie dieses Dialogfeld über die Menübefehle **Extras | Anpassen.**
2. Klicken Sie auf den Menüeintrag oder das Symbol, welches Sie löschen möchten, und ziehen Sie es mit der Maus an eine Stelle, an der die kleine Schaltfläche am Zeiger angezeigt wird.
3. Lassen Sie die Maustaste los, und der Befehl wird entfernt.

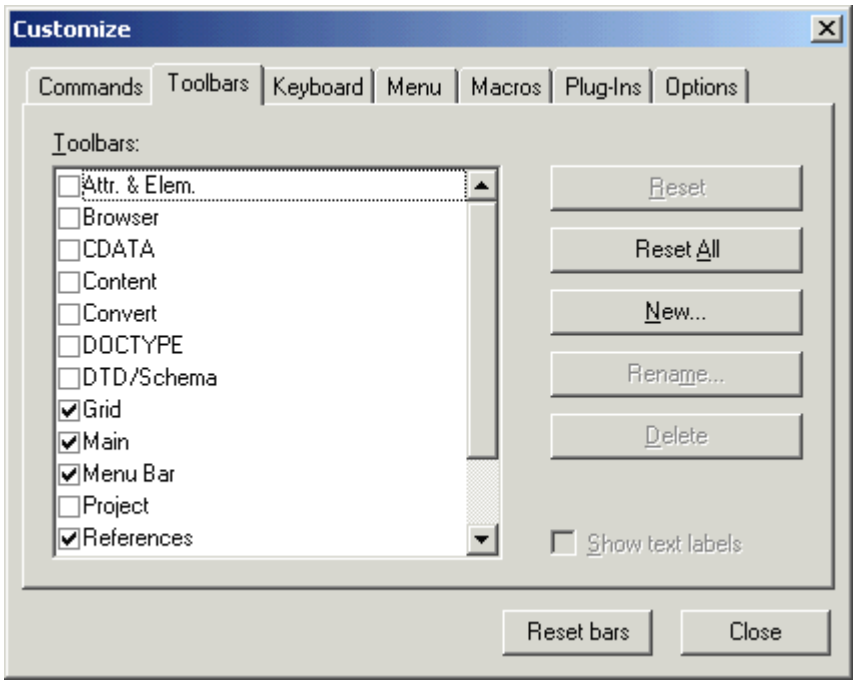
Symbolleisten

Auf dem Register Symbolleisten können Sie spezifische Symbolleisten aktivieren oder deaktivieren. Weiters können Sie eigene Symbolleisten erstellen.

XMLSpy Symbolleisten enthalten Symbole für die meistverwendeten Befehle.
Zu jedem Symbol gehört ein kurzer Tooltip, der angezeigt wird, wenn Sie mit dem Cursor auf das Symbol gehen. In der Statusleiste wird eine detailliertere Beschreibung zum jeweiligen Symbol angezeigt.

Sie können die Symbolleisten aus Ihrer Standardposition an eine andere beliebige Stelle am Bildschirm verschieben, sowie diese an den rechten oder linken Rand des Hauptfensters andocken.

- Symbolleisteneinstellungen, die in der Grid-, Schema/WSDL Design- oder Text-Ansicht definiert wurden, stehen in den jeweiligen Ansichten zur Verfügung. Die Symbolleisten der Browser-Ansicht sind unabhängig von der jeweiligen Ansicht immer verfügbar.



So aktivieren oder deaktivieren Sie eine Symbolleiste:

1. Klicken Sie auf das Kontrollkästchen um die jeweilige Symbolleiste zu aktivieren oder deaktivieren.

So erstellen Sie eine neue Symbolleiste:

1. Klicken Sie auf die Schaltfläche **Neu...** und geben Sie der neuen Symbolleiste einen Namen (Dialogfeld Symbolleistenname).
2. Fügen Sie Befehle in die neue Symbolleiste ein (Register Befehle im Dialogfeld **Anpassen**).

So setzen Sie die Menüleiste zurück

- Klicken Sie auf den Menüleisteneintrag und
- Klicken Sie auf die Schaltfläche **Zurücksetzen**, um die Menübefehle auf Werkseinstellung zurückzusetzen.

So setzen Sie alle Symbolleisten und Menübefehle zurück

- Klicken Sie auf die Schaltfläche **Alles Zurücksetzen**, um die Befehle der Symbolleisten auf Werkseinstellung zurückzusetzen. Sie erhalten eine Meldung, dass nun alle Symbolleisten und Menüs zurückgesetzt werden.
- Klicken Sie auf Ja um das Zurücksetzen zu bestätigen.

So ändern Sie den Namen einer Symbolleiste:

- Klicken Sie auf die Schaltfläche **Umbenennen...** um den Namen der Symbolleiste zu ändern.

So löschen Sie eine Symbolleiste:

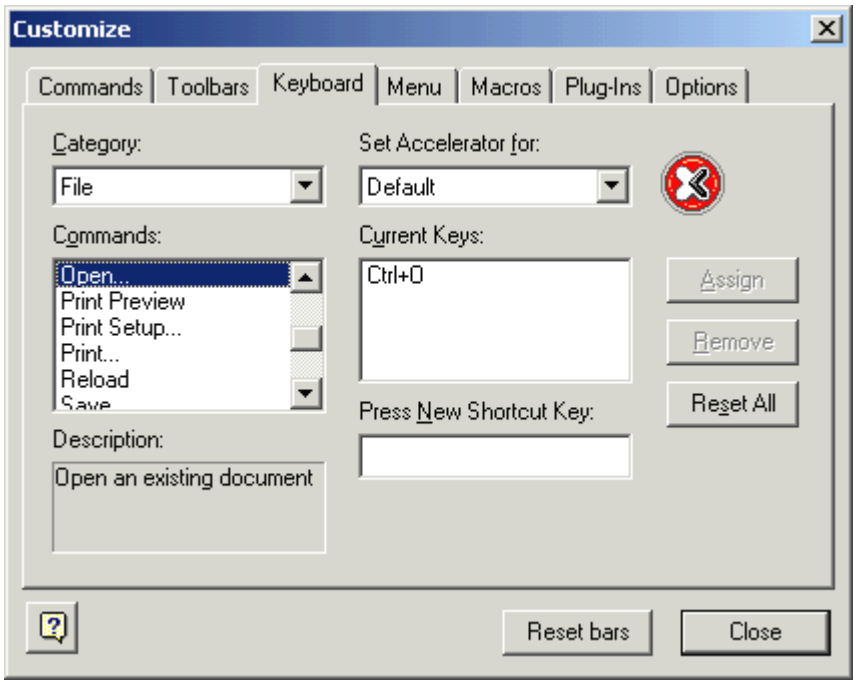
- Klicken Sie auf die Schaltfläche **Löschen** um die im Listenfeld markierte Symbolleiste zu löschen.
- Sie erhalten eine Meldung, in der Sie gefragt werden, ob die markierte Symbolleiste tatsächlich gelöscht werden soll. Bestätigen Sie den Löschvorgang mit Ja.
- oder, rechtsklicken Sie auf die Menü- oder Symbolleiste und wählen Sie Löschen aus dem Popup-Menü.

© 2007 Altova GmbH

Textlabels zeigen:
Wenn Sie diese Option aktivieren, werden Hilfetexte zu den Symbolen angezeigt.

Tastatur

Auf dem Register Tastatur können Sie Tastaturkürzel für jeden Befehl in XMLSpy definieren (oder löschen).



So weisen Sie einem Befehl ein neues Kürzel zu:

1. Wählen Sie die Befehlskategorie im Dropdown-Listenfeld **Kategorie**.
2. Wählen Sie den **Befehl** aus dem Listenfeld, welchem Sie ein neues Kürzel zuweisen möchten.
3. Klicken Sie in das "Tastaturkürzel eingeben:" Textfeld und geben Sie das Kürzel ein. Die Kürzel werden sofort im Textfeld angezeigt. Wenn das Kürzel schon einem anderen Befehl zugeordnet war, wird diese Funktion im Textfeld angezeigt.
4. Klicken Sie auf die Schaltfläche **Zuweisen**.
5. Das neue Kürzel wird in die Liste der aktuellen Kürzel aufgenommen. (um das Textfeld zu **löschen** drücken Sie die Strg-, ALT- oder Umschalttaste).

So heben Sie eine Zuweisung auf (oder löschen ein Kürzel):

1. Klicken Sie auf den Kürzel in der Liste der aktuellen Kürzel, der gelöscht werden soll und
2. Klicken Sie auf die Schaltfläche **Entfernen**, die jetzt aktiv ist.
3. Klicken Sie auf die Schaltfläche Schließen um die Änderungen zu speichern.

Kürzel setzen für:
Derzeit noch keine Funktion.

Zugewiesene Tastatur-Kürzel:

Kürzel nach Tasten

F1	Hilfe-Menü
F3	Weitersuchen
F5	Aktualisieren
F7	Wohlgeformtheit prüfen
F8	Validieren
F9	Breakpoint einfügen/löschen
Umschalt+F9	Tracepoint einfügen/löschen
Strg+F9	Breakpoint aktivieren/deaktivieren
Umschalt+Strg+F9	Tracepoint aktivieren / deaktivieren
F10	XSL Transformation
Strg+F10	XSL:FO Transformation
F11	Einsteigen
Strg+F11	Überspringen
Umschalt + F11	Aussteigen
Alt+F11	Debugger starten
Num +	Erweitern
Num -	Reduzieren
Num *	Alles erweitern
Strg+Num-	Unselektierte reduzieren
Strg + G	Gehe zu Zeichen/Zeile
Strg+TAB	Wechselt zwischen offenen Dokumenten
Strg+F6	Offene Fenster durchblättern
Pfeiltasten oben/unten	Laufleiste auf/ab
Esc.	Dialogfeld schließen/Editieren abbrechen
Eingabetaste/Leerzeichentaste	Bestätigen
Alt + F4	Schließt XMLSpy
Strg + F4	Schließt aktives Fenster
Alt + F, 1	Öffnet letzte Datei
Strg + Doppelklick Element (Schema-Ansicht)	Elementdefinition anzeigen
Strg + N	Datei Neu
Strg + O	Datei Öffnen
Strg + S	Datei Speichern
Strg + P	Datei Drucken
Strg + A	Alles auswählen
Umschalt + Del	Ausschneiden (oder Strg + X)
Strg + C	Kopieren
Strg + V	Einfügen
Strg + Z	Rückgängig
Strg + Y	Wiederherstellen
Del	Löschen (Objekt in der Schema-Ansicht/Enhanced Grid Ansicht)

© 2007 Altova GmbH

Strg + F	Suchen
F3	Weitersuchen
Strg + H	Ersetzen
Strg + I	Attribut anhängen
Strg + E	Element anhängen
Strg + T	Text anhängen
Strg + D	CDATA anhängen
Strg + M	Comment anhängen
Strg + Umschalt + I	Attribut einfügen
Strg + Umschalt + E	Element einfügen
Strg + Umschalt + T	Text content einfügen
Strg + Umschalt + D	CDATA einfügen
Strg + Umschalt + M	Comment einfügen
Strg + ALT + I	Child Attribute hinzu
Strg + ALT + E	Child Element hinzu
Strg + ALT + T	Child Text hinzu
Strg + ALT + D	Child CDATA hinzu
Strg + ALT + M	Child Comment hinzu

Tastaturkürzel für die Textansicht

Strg + "+"	Vergrößern
Strg + "-"	Verkleinern
Strg + 0	Zoom zurücksetzen
Strg + Mausrad nach vorne	Vergrößern
Strg + Mausrad nach hinten	Verkleinern

Kürzel nach Funktion

Editieren abbrechen	Esc.
Child Attribute hinzu	Strg + ALT + I
Child CDATA hinzu	Strg + ALT + D
Child Comment hinzu	Strg + ALT + M
Child Element hinzu	Strg + ALT + E
Child Text hinzu	Strg + ALT + T
Attribut anhängen	Strg + I
CDATA anhängen	Strg + D
Comment anhängen	Strg + M
Element anhängen	Strg + E
Text anhängen	Strg + T
Wohlgeformtheit prüfen	F7
Aktives Fenster schließen	Strg + F4
Schließt XMLSpy	Alt + F4
Reduzieren	Num -
Unselektierte reduzieren	Strg + Num-
Bestätigen	Return / Space bar
Kopieren	Strg + C
Ausschneiden	Umschalt + Del (or Strg + X)
Offenen Fenster durchblättern	Strg + TAB und Strg + F6
Löschen	Del
Breakpoint aktivieren/deaktivieren	Strg + F9
Tracepoint aktivieren/deaktivieren	Umschalt + Strg + F9
Erweitern	Num +
Alles erweitern	Num *
Datei Neu	Strg + N
Datei Öffnen	Strg + O
Datei Drucken	Strg + P
Datei Speichern	Strg + S
Suchen	Strg + F
Weitersuchen	F3
Gehe zu Zeichen/Zeile	Strg + G
Hilfemenü	F1
Attribut einfügen	Strg + Umschalt + I
CDATA einfügen	Strg + Umschalt + D
Comment einfügen	Strg + Umschalt + M
Element einfügen	Strg + Umschalt + E
Breakpoint einfügen/löschen	F9
Tracepoint einfügen/löschen	Umschalt+F9
Text content einfügen	Strg + Umschalt + T
Laufleiste auf/ab	Pfeiltasten oben/unten
Letzte Datei öffnen	Alt + F, 1
Einfügen	Strg + V
Wiederherstellen	Strg + Y
Aktualisieren	F5
Ersetzen	Strg + H
Alles markieren	Strg + A
Start Debugger	Alt + F11
Einsteigen	F11
Aussteigen	Umschalt + F11
Überspringen	Strg + F11
Definition anzeigen	Strg + Doppelklick Element

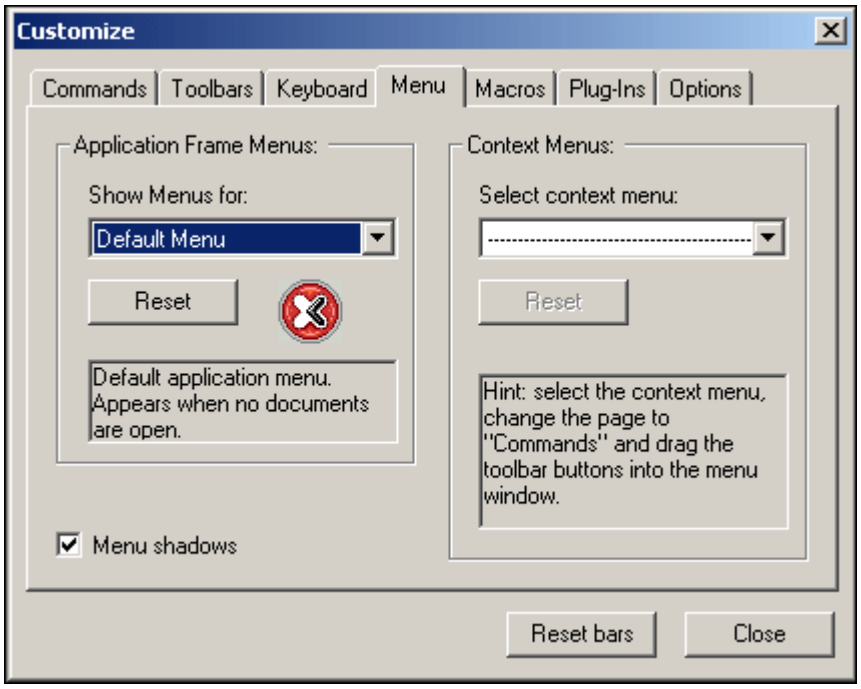
© 2007 Altova GmbH

Rückgängig	Strg + Z
Validieren	F8
XSL Transformation	F10
XSL:FO Transformation	Strg + F10

In der Applikation sehen Sie im Dialogfeld "Tastaturhilfe" (**Hilfe | Tastaturzuordnungen**) eine Liste der Befehle mit Ihren Tastaturkürzeln und Beschreibungen.

Menü

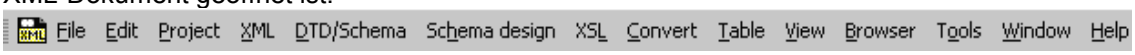
Auf dem Register Menü können Sie die Menüleisten und die Rechtsklick-Kontextmenüs anpassen.



Sie können sowohl die standardmäßigen Menüleisten, als auch die XMLSpy Menüleisten anpassen.

Das **Standardmenü** wird angezeigt, wenn Sie kein XML-Dokument in XMLSpy geöffnet haben.

Das XMLSpy-Menü ist die Menüleiste, die dargestellt wird, wenn mindestens ein XML-Dokument geöffnet ist.



So passen Sie ein Menü an:

1. Wählen Sie die Menüleiste aus dem Dropdown-Listenfeld "Menüs anzeigen für:", welche Sie anpassen wollen.
2. Klicken Sie auf das Register **Befehle** und ziehen Sie die gewünschten Befehle in das ausgewählte Menü.

So löschen Sie Befehle aus einem Menü:

1. Rechtsklicken Sie bei geöffnetem Dialogfeld "Anpassen" auf den Befehl, oder das Symbol, welches für den Befehl steht, und
2. Wählen Sie die Option Löschen aus dem Popup Menü,

oder

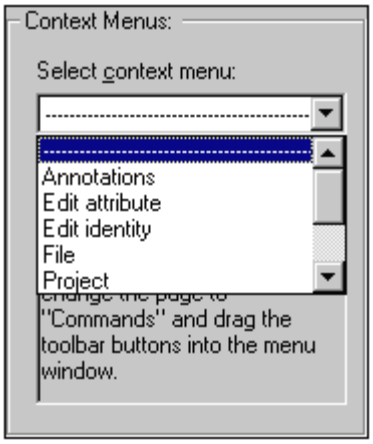
1. Wählen Sie **Extras | Anpassen** um das Dialogfeld Anpassen zu öffnen und
2. Ziehen Sie den Befehl aus dem Menü bis am Mauszeiger die kleine Schaltfläche angezeigt wird.

So setzen Sie eine der Menüleisten zurück:

1. Wählen Sie entweder den Eintrag Standard oder XMLSpy aus dem Dropdown-Listenfeld und
2. Klicken Sie auf die Schaltfläche **Zurücksetzen** unter dem Menünamen. Mit einer Meldung werden Sie gefragt, ob Sie die Menüleiste tatsächlich zurücksetzen möchten.

So passen Sie ein Kontextmenü an (Rechtsklick-Menü):

1. Wählen Sie das entsprechende Kontextmenü aus dem Dropdown-Listenfeld.
2. Klicken Sie auf das Register **Befehle** und ziehen Sie die Befehle in das offene Kontextmenü.



So löschen Sie Befehle aus einem Kontextmenü:

1. Rechtsklicken Sie bei geöffnetem Dialogfeld "Anpassen" auf den Befehl, oder das Symbol, welches für den Befehl steht, und
2. Wählen Sie die Option Löschen aus dem Popup Menü,

oder

1. Wählen Sie **Extras | Anpassen** um das Dialogfeld Anpassen zu öffnen und
2. Ziehen Sie den Befehl aus dem Menü heraus und lassen Sie die Maustaste los, sobald am Mauszeiger die kleine Schaltfläche angezeigt wird.

So setzen Sie eines der Kontextmenüs zurück:

1. Wählen Sie das Kontextmenü aus dem Dropdown-Listenfeld und
2. Klicken Sie auf die Schaltfläche **Zurücksetzen** unter dem Menünamen. Mit einer Meldung werden Sie gefragt, ob Sie das Kontextmenü tatsächlich zurücksetzen möchten.

So schließen Sie ein Kontextmenüfenster:

1. Klicken Sie auf das Symbol **Schließen** rechts oben in der Titelleiste oder
2. Klicken Sie auf die Schaltfläche "Schließen" im Dialogfeld "Anpassen".

Menüanimationen (vor Windows 2000)

- Wählen Sie eine der Menüanimationen aus dem Dropdown-Listenfeld, wenn Sie animierte Menüs sehen möchten. Bitte beachten Sie, dass das Dropdown-Listenfeld nur für Windows-Versionen vor Windows 2000 verfügbar ist. Ab Windows 2000 können

© 2007 Altova GmbH

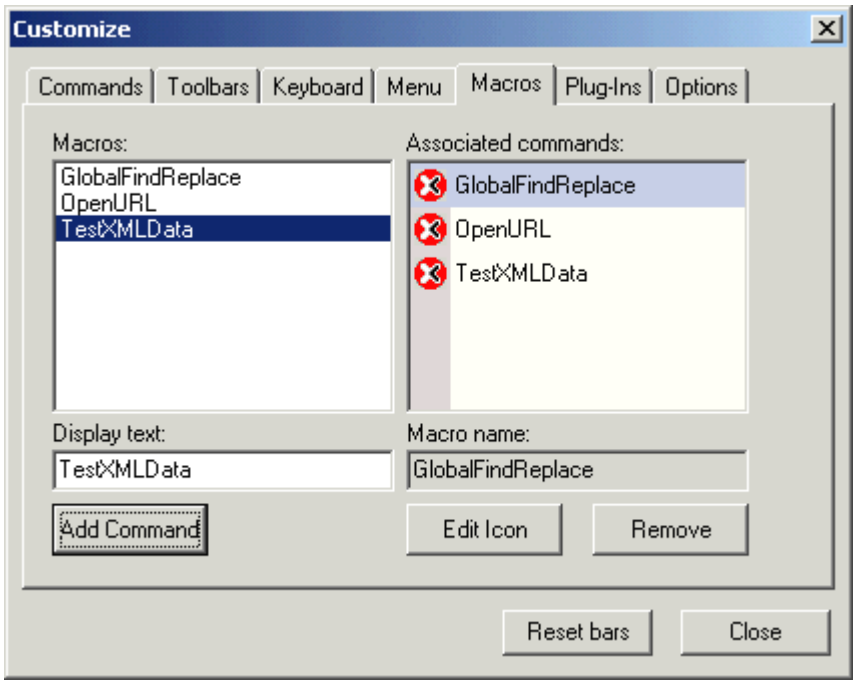
die Einstellungen auf dem Register Effekte im Dialogfeld "Eigenschaften von Anzeige" geändert werden. Doppelklicken Sie in der Systemsteuerung auf das Symbol **Anzeigen** , um das Dialogfeld zu öffnen.

Menüschatten

- Klicken Sie auf das Kontrollkästchen Menüschatten, wenn Menüs mit Schatten dargestellt werden sollen.

Makros

Auf dem Register Makros können Sie Makros, die Sie zuvor in der XML Skript-Umgebung erzeugt wurden, in eine Symbolleiste oder ein Menü einfügen.



So fügen Sie das Makrosymbol in eine Symbolleiste oder ein Menü ein:

1. Starten Sie die Skriptumgebung über **Extras | Skripting-Umgebung**.
2. Doppelklicken Sie auf den Eintrag XMLSpyMacros im Ordner Module des Projekts **XMLSpyGlobalScripts**.
   Zuvor definierte Makros werden im rechten Fenster angezeigt.
3. Wechseln Sie zurück in XMLSpy und wählen Sie **Extras | Anpassen** und klicken Sie auf das Register Makros.
   Die in der Skriptumgebung definierten Makros werden nun im Makros Listenfeld links angezeigt.
4. Klicken Sie auf den Namen des Makros und dann die Schaltfläche "Befehl hinzufügen". Der Makroname wird im Listenfeld "Zugewiesene Befehle" angezeigt.
5. Klicken Sie im Listenfeld "Zugewiesene Befehle" auf den Namen des Makros und ziehen Sie ihn in die Symbolleiste oder das Menü.

- Zum Bearbeiten des Symbols für ein Makro, klicken Sie auf die Schaltfläche **Symbol bearbeiten**.

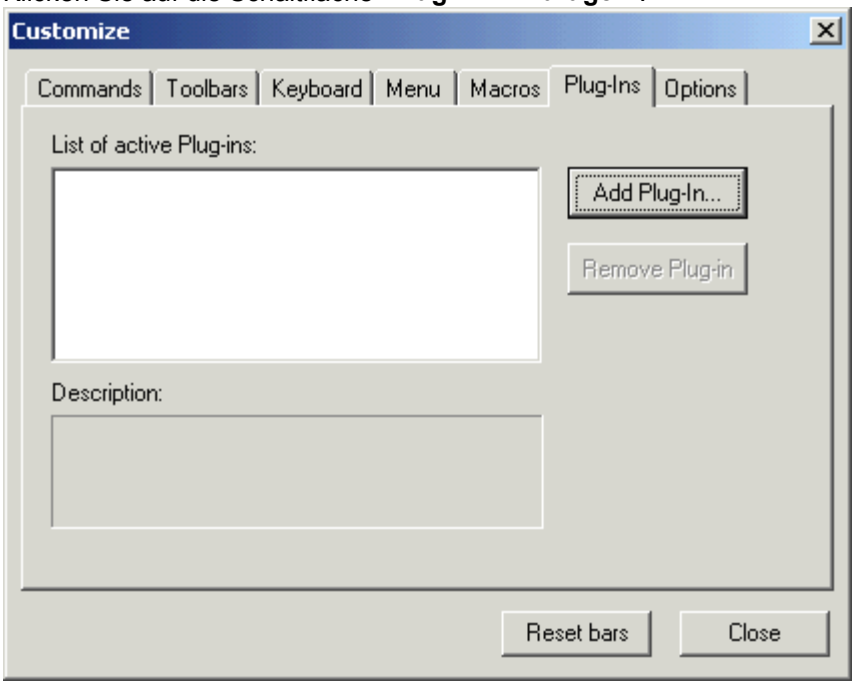
- Um ein Makro aus dem Listenfeld "Zugewiesene Befehle" zu löschen, klicken Sie auf die Schaltfläche Entfernen.

Plug-Ins

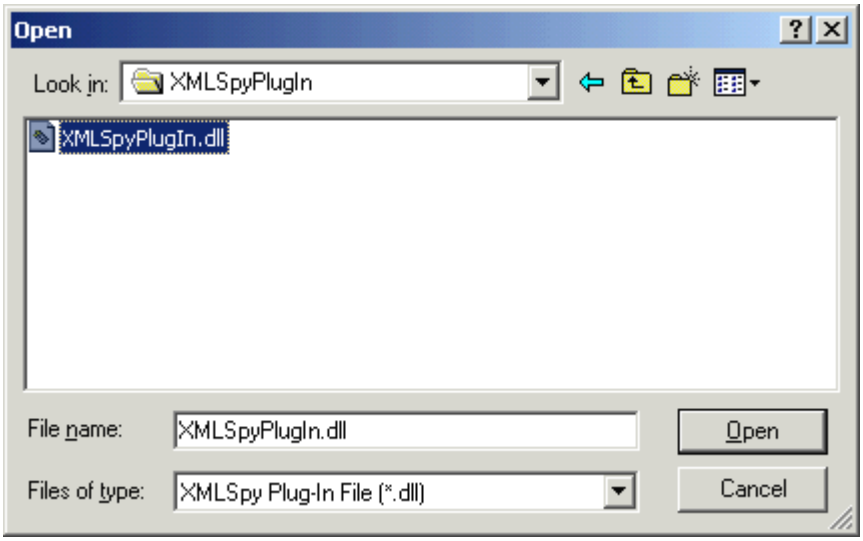
Auf dem Register Plug-Ins können Sie Plug-Ins in eine Symbolleiste oder ein Menü einfügen.

So fügen Sie das Symbol für ein Plug-In in eine Symbolleiste oder Menü ein:

1. Klicken Sie auf die Schaltfläche "**Plug-In hinzufügen**".



2. Wählen Sie den Ordner aus und klicken Sie auf die Plug-In Datei um diese zu markieren (in diesem Beispiel: XMLSpyPlugIn.dll).



3. Klicken Sie auf die Schaltfläche **Öffnen** um das Plug-In zu installieren.
   Der Name des Plug-Ins wird in der Liste "Derzeit aktive Plug-Ins" angezeigt und das Symbol des/der Plug-Ins wird in einer neuen Symbolleiste angezeigt.

So entfernen Sie ein Plug-In:

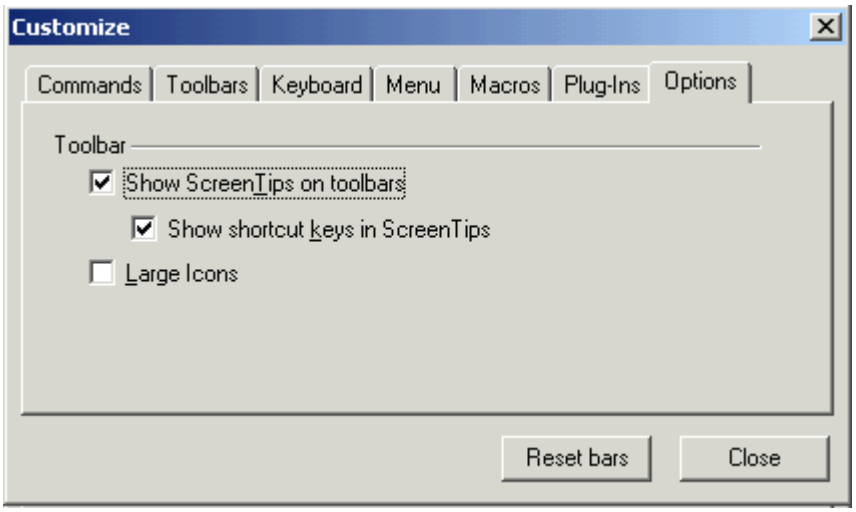
1. Klicken Sie auf den Namen des Plug-Ins in der Liste "Derzeit aktive Plug-Ins" und

© 2007 Altova GmbH

klicken Sie auf die Schaltfläche "Plug-In löschen".

Optionen

Auf dem Register Optionen können Sie verschiedene, allgemeine Einstellungen vornehmen.



Symbolleiste
Wenn das Kontrollkästchen **Tooltip in Symbolleiste einblenden** aktiviert ist, wird ein Popup beim Ziehen des Mauszeigers über das entsprechende Symbol in der Symbolleiste angezeigt. Das Popup enthält eine Kurzbeschreibung der Funktion, sowie, falls vorhanden, ein Tastaturkürzel.

Das Kontrollkästchen "Tastaturkürzel in Tooltips anzeigen" ermöglicht es Ihnen, auszuwählen, ob die Tastaturkürzel (wo vorhanden) angezeigt werden sollen.

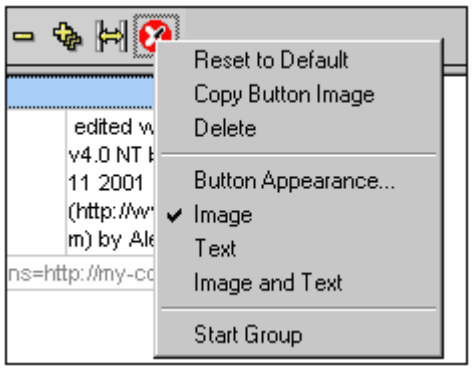
Mit dem Kontrollkästchen Große Symbole können Sie die Größe der angezeigten Symbole beeinflussen.

Kontextmenü 'Anpassen'

Über das Kontextmenü "Anpassen" können Sie Symbole und Menüobjekte Ihren Ansprüchen entsprechend anpassen.

So öffnen Sie das Kontextmenü "Anpassen":

1. Zuerst öffnen Sie das Menü Anpassen über den Menübefehl **Extras | Anpassen**.
2. Dann setzen Sie den Mauszeiger auf das Symbol (oder Menü) und **rechtsklicken Sie**, um das Kontextmenü zu öffnen.



Zurücksetzen
Derzeit keine Funktion.

Schaltflächensymbol kopieren
Das Symbol wird in die Zwischenablage kopiert.

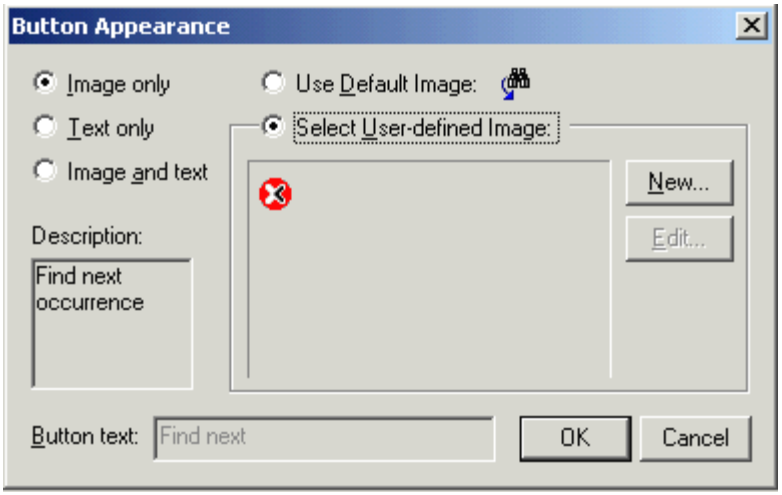
Löschen
Mit dieser Option wird das Symbol oder Menü gelöscht, auf welches Sie zuvor rechtsgeklickt haben. Wenn Sie ein Menü löschen, werden alle enthaltenen Menüoptionen gelöscht!

So stellen Sie ein gelöschtes Menü wieder her:

- Wählen Sie das Register Menü
- Wählen Sie das Menü, welches wiederhergestellt werden soll (XMLSpy oder Standardmenü).
- Klicken Sie auf die Schaltfläche Zurücksetzen unter dem Menü im Dropdown-Listenfeld.

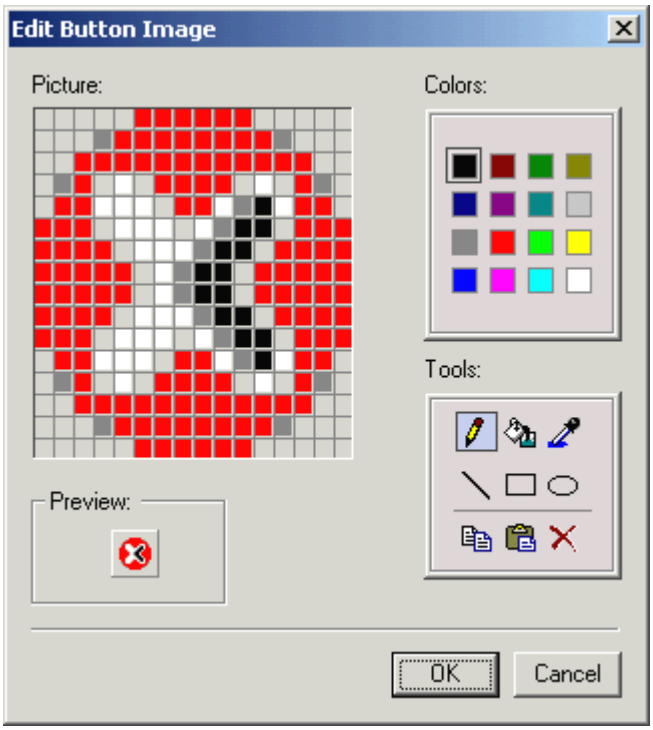
Schaltflächendarstellung
Mit dieser Option können Sie Schaltflächensymbole und Text bearbeiten. Derzeit kann nur das Symbol für Makros bearbeitet werden (Standardsymbol XMLSpy).

Mit den Optionsfeldern "Nur Symbol", "Nur Text" oder "Symbol und Text" können Sie definieren, was Sie editieren möchten. Klicken Sie auf das Optionsfeld **Benutzerdefiniertes Symbol** und klicken Sie auf eines der verfügbaren Symbole.



Klicken Sie auf die Schaltfläche "Bearbeiten" um das Dialogfeld **Schaltflächensymbol editieren** zu öffnen. Der Schaltflächentext kann bearbeitet werden, wenn Sie entweder die Option "Nur Text" oder "Symbol und Text" ausgewählt haben.

© 2007 Altova GmbH



Symbol
Mit dieser Option ändern Sie das graphische Design des Symbols.

Text
Mit dieser Option ändern Sie die Textbeschreibung.

Symbol und Text
Mit dieser Option wird die gleichzeitige Darstellung des Symbols und des Textes aktiviert.

Gruppe starten
Mit dieser Option wird links neben dem Symbol, auf welches Sie rechtsgeklickt haben, ein vertikaler Trennungsstrich angezeigt.

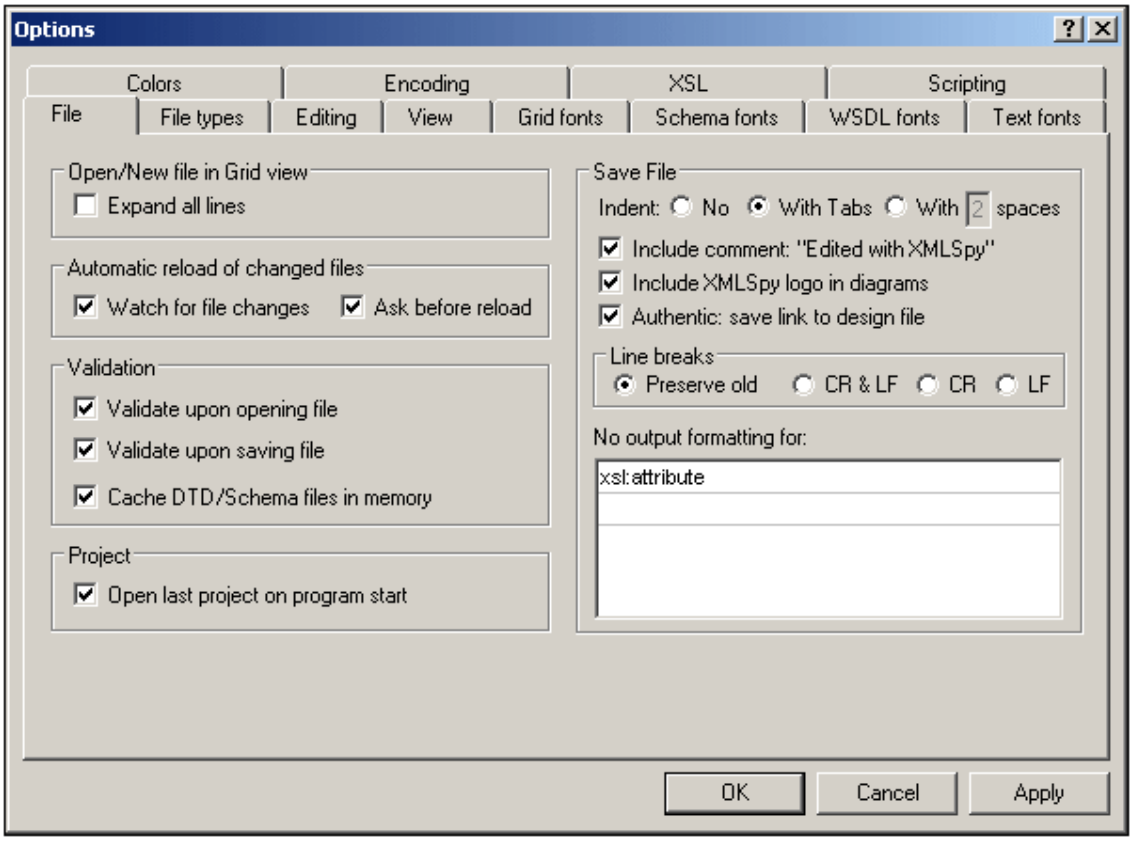
15.15.8 Optionen

Mit dem Befehl "Optionen" können Sie globale Programmeinstellungen in einem mit Registern versehenen Dialogfeld vornehmen. Diese Einstellungen werden in der Registry gespeichert und auf alle aktuellen und zukünftigen Dokumentfenster angewendet. Die Schaltfläche "Übernehmen" zeigt die Änderungen in den aktuell geöffneten Dokumenten an und speichert die Einstellungen. Die Änderungen sehen Sie sofort in den Fenstern im Hintergrund.

In diesem Abschnitt werden die einzelnen Register des Dialogfelds "Optionen" beschrieben.

Datei

Das Register "**Datei**" definiert die Art, wie XMLSpy Ihre XML-Dokumente öffnet und speichert. Weitere Einstellungen hierzu finden Sie auf dem Register Kodierung.



Öffnen/Neue Datei in Grid-Ansicht
Sie können festlegen, ob eine existierende oder eine neue Datei entweder in der Enhanced Grid- oder der Text-Ansicht geöffnet werden soll. Zusätzlich können Sie beim Auswählen der Grid-Ansicht auswählen, ob alle Zeilen ausgeklappt werden sollen.

Auto-laden von geänderten Dateien
Wenn Sie in einer Multi-User Umgebung, oder mit Dateien arbeiten, die von Servern dynamisch erstellt werden, können Sie die Änderungen während des Bearbeitens direkt verfolgen. XMLSpy wird Sie fragen, ob die Datei neu geladen werden soll, sobald eine Änderung in einem offenen Dokument entdeckt wurde.

Validierung
Wenn Sie Dokument Type Definitions (DTDs) oder Schemas zur Definition der Struktur Ihrer XML-Dokumente verwenden, können Sie das Dokument beim Öffnen oder Speichern automatisch auf Gültigkeit überprüfen lassen.
Mit XMLSpy können Sie die Dateien auch in den Speicher cachen, sodass unnötiges Neuladen entfällt (z.B. wenn das referenzierte Schema über eine URL aufgerufen wird). Falls in Ihrer schemaLocation-Deklaration eine URL verwendet wird, deaktivieren Sie die Option "DTD/Schema Dateien im Cachespeicher", um am Schema vorgenommene Änderungen sofort anzuzeigen und nicht die im Cache gespeicherte Version des Schemas zu verwenden.

Projekt
Beim Starten von XMLSpy können Sie das zuletzt verwendete Projekt automatisch anzeigen lassen.

Datei speichern
Beim Speichern einer Datei werden die Einzüge so wie in diesem Fenster angegeben

© 2007 Altova GmbH

gespeichert, d.h. mit TAB-Zeichen (#x09), zwei Leerzeichen oder ohne Einzug. Wenn ein StyleVision Power Stylesheet mit einer XML-Datei verknüpft ist, bewirkt die Option "Authentic: Link zu Design-Datei speichern", dass die Verknüpfung zum StyleVision Power Stylesheet mit der XML-Datei gespeichert wird. Beachten Sie, dass der hier defnierte Einzug auch verwendet wird, wenn Sie das Dokument mit Hilfe der Funktion "Pretty-Print" formatieren (**Bearbeiten | Pretty-Print**).

Beim Speichern eines XML-Dokuments fügt XMLSpy im oberen Bereich des Dokuments den kurzen Kommentar `<!-- Mit XMLSpy bearbeitet http://www.altova.com -->` ein. Dieser Kommentar kann nur von lizenzierten Benutzern deaktiviert werden und wird beim Bearbeiten oder Speichern von Dateien in der Enhanced Grid-Ansicht oder der Schema Design-Ansicht verwendet.

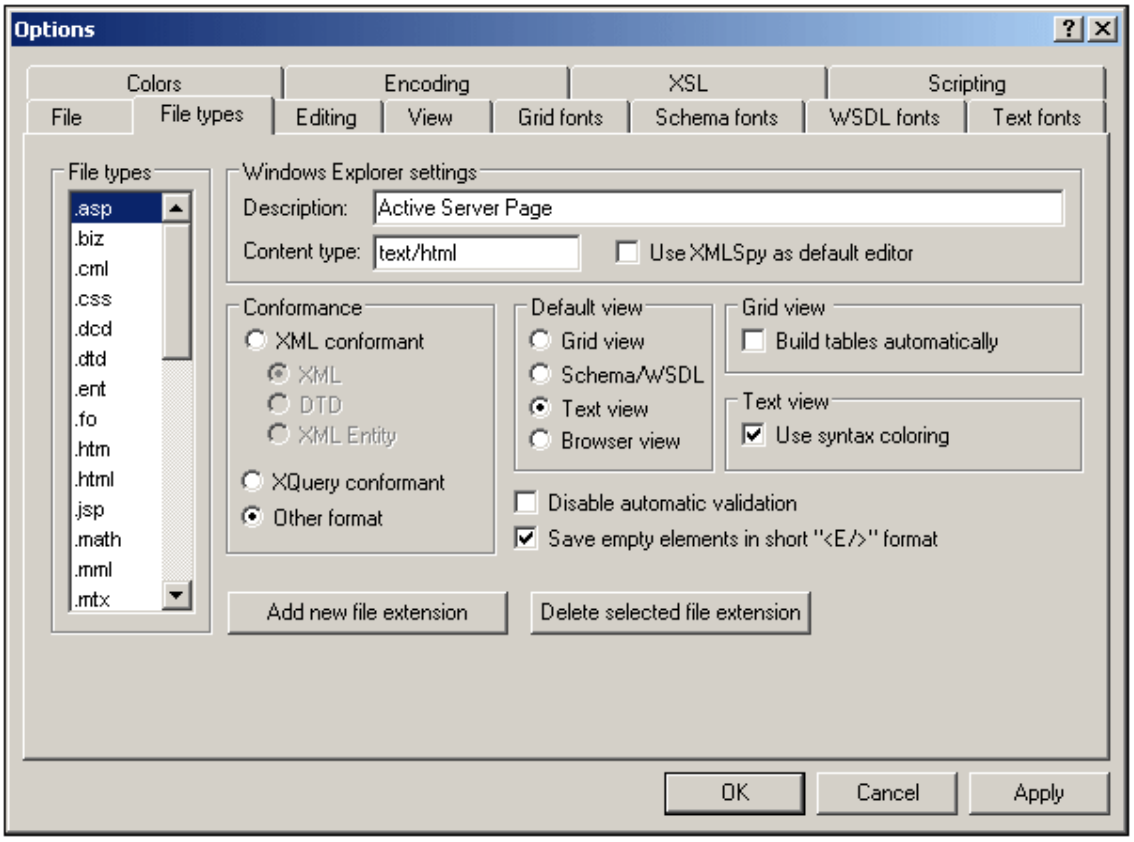
Wenn Sie ein Content Model-Diagramm speichern (über die Menüoption **Schema-Design | Schema-Dokumentation generieren**), inkludiert XMLSpy das XMLSpy Logo. Diese Option kann nur von lizenzierten Benutzern deaktiviert werden.

Zeilenumbrüche
Wenn Sie eine Datei öffnen, werden die Zeichen für Zeilenumbrüche beibehalten, wenn Sie die Option **Alte beibehalten** gewählt haben. Alternativ dazu können Zeilenumbrüche in **CR&LF** für PCs, **CR** für MacOS, oder **LF** für Unix kodiert werden.

Keine Ausgabe-Formatierung für
In der Textansicht können Sie Child-Elemente aus Gründen der besseren Übersichtlichkeit einrücken (mit Tabulator oder Leerzeichen). Wenn Sie einen Einzug definiert haben, wird die Datei mit den jeweiligen Whitespaces gespeichert. Der Einzug eines Elements dient auch dazu, seine Position in der Dokumentenhierarchie deutlich zu machen (**Bearbeiten | Pretty-Print**). Sie können jedoch den (in diesem Dialogfeld oder manuell in der Datei) definierten Einzug für einzelne Elemente auch außer Kraft setzen, indem Sie den Namen des Elements in das Feld **Keine Ausgabe-Formatierung für** eingeben. Alle in diesem Feld definierten Elemente werden ohne Whitespaces zwischen aufeinanderfolgenden untergeordneten Elementen gespeichert (und angezeigt).

Dateiarten

Auf dem Register "Dateiarten" können Sie für XMLSpy verschiedene Einstellungen für die einzelnen Dateiarten festlegen.



Wählen Sie eine Dateiart im Listenfeld "Dateiarten" aus, um folgende Funktionen für die ausgewählte Dateiart anzupassen:

Windows Explorer Einstellungen
Sie können die Beschreibug der Dateiart und den MIME-konformen Inhaltstyp, der von Windows Explorer verwendet wird, ändern. Weiters können Sie definieren, ob XMLSpy als der standardmäßige Editor für diese Dateiart verwendet werden soll.

XML Konformität

XMLSpy bietet für verschiedene Dateiarten spezifische Editierfunktionen und andere Funktionen, die Sie durch Einstellung der Konformität in dieser Option festlegen können. In XMLSpy können Sie für eine Dateiart festlegen, dass sie mit XML-, XQuery und anderer (Text)-Syntax konform sein muss. Des Weiteren lässt sich für XML-Dateiarten die Konformität mit XML, DTD and XML Entitäten festlegen. Für viele Dateiarten ist bereits eine standardmäßige Konformitätseinstellung festgelegt, die am besten beibehalten werden sollte, es sei denn Sie fügen eine neue Dateiart hinzu oder möchten für eine bestimmte Dateiart die Konformität explizit ändern.

Standard Ansicht
In dieser Gruppe definieren Sie, welche Standardansicht für welche Dateiart verwendet werden soll.

Grid-Ansicht
Dieses Kontrollkästchen definiert, ob in der Enhanced Grid Ansicht automatisch Tabellen erzeugt werden sollen.

Textansicht

 © 2007 Altova GmbH

In diesem Textfeld können Sie die Syntaxfärbung für bestimmte Dateiarten bestimmen.

Automatische Validierung deaktivieren
Mit dieser Option können Sie für bestimmte Dateiarten die automatische Validierung deaktivieren. Die automatische Validierung wird normalerweise beim Öffnen oder Speichern einer Datei oder beim Wechseln der Ansicht durchgeführt.

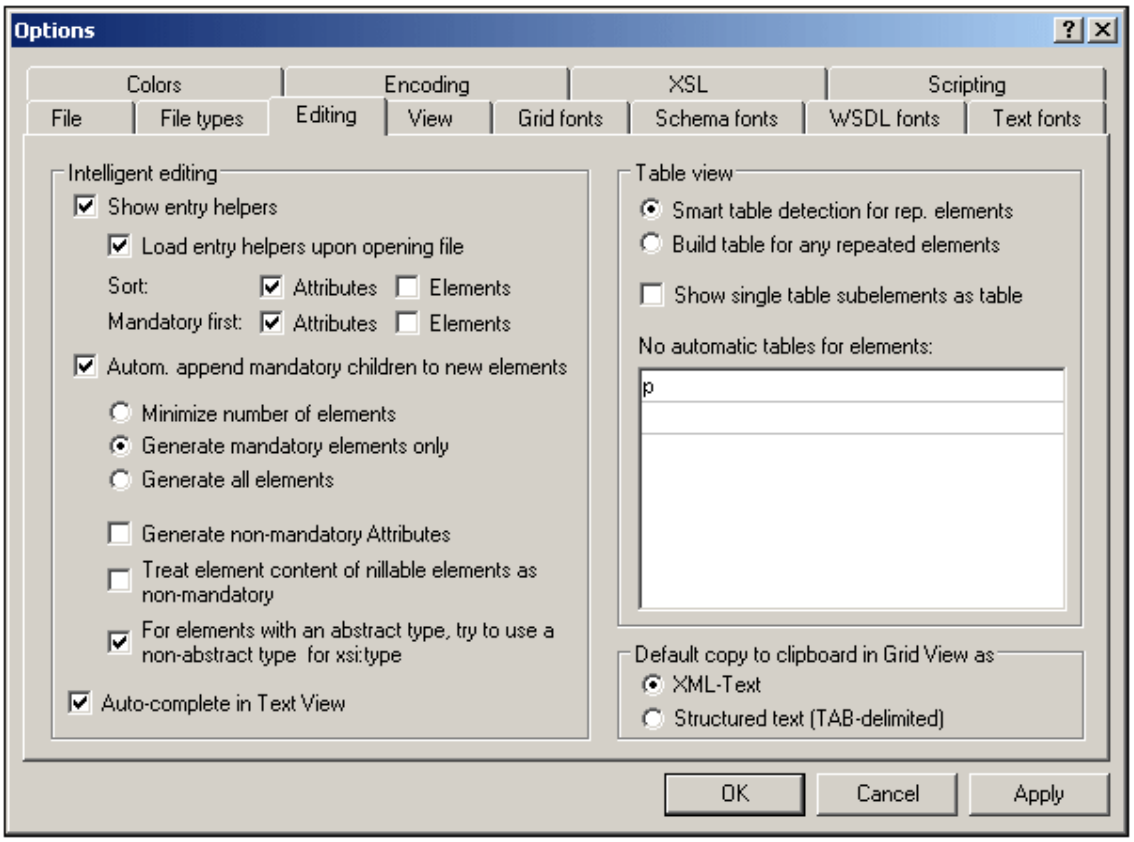
Leere Elemente im Kurzformat "<E/>" speichern
Manche XML-Tools haben Probleme, die Kurzform <Element/> für leere Elemente zu verstehen (die in der XML 1.0 Spezifikation definiert wurde). Sie können XMLSpy anweisen, diese Elemente in der Langform zu speichern <Element></Element> , die gleichermaßen gültig ist.

Schaltfläche "Dateierweiterung hinzufügen"
Eine neue Dateierweiterung wird in die Liste der Dateiarten aufgenommen. Mit den Optionen in diesem Dialogfeld definieren Sie die Einstellungen für die neue Dateiart.

Schaltfläche "Markierte Dateierweiterung löschen"
Löscht die Dateiart und die dazugehörigen Einstellungen.

Bearbeiten

Auf dem Register "**Bearbeiten**" können Sie Bearbeitungseinstellungen für XMLSpy definieren.



Editier-Hilfen
XMLSpy bietet Ihnen intelligente Eingabehilfen zum Bearbeiten von Dokumenten. Verschiedene Bearbeitungseinstellungen für diese Eingabehilfen lassen sich hier anpassen.

In die Zwischenablage der Grid-Ansicht kopieren als

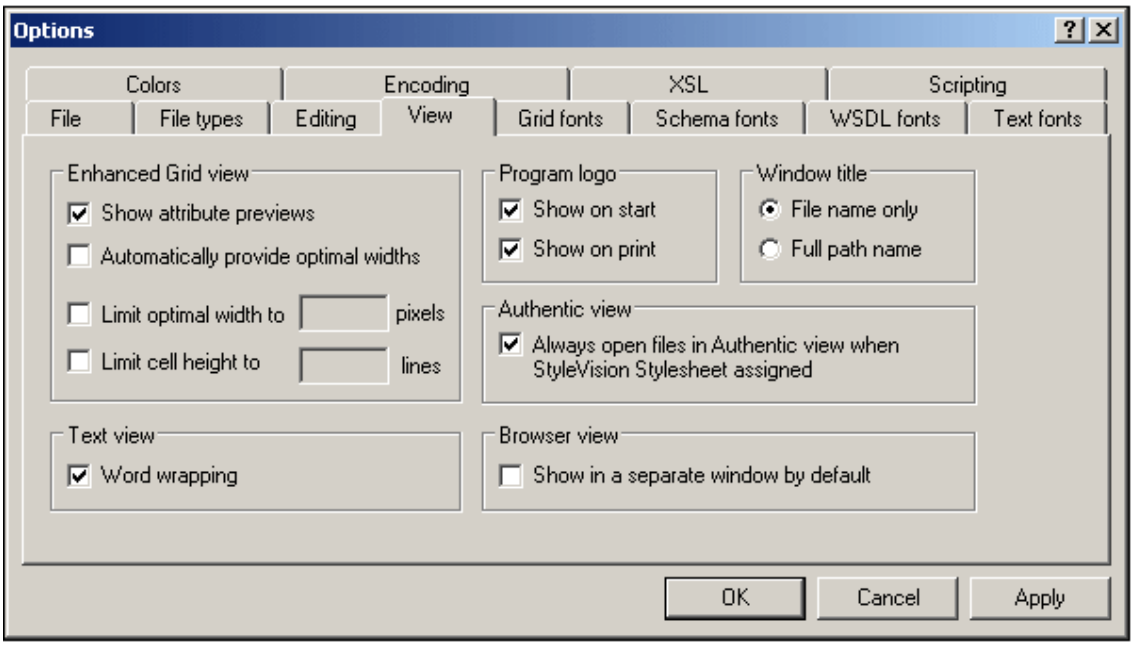
Wählen Sie das Format, in dem Daten in Fremdanwendungen exportiert werden sollen. Wenn Sie XML-Text wählen, wird der Inhalt der Zwischenablage genau wie die resultierende XML-Datei formatiert und mit Tags versehen.

Im strukturierten Textmodus wird versucht, den Inhalt der Zwischenablage als Tabelle (Verwendung in Spreadsheet- oder Datenbankapplikationen) zu formatieren. Diese Option betrifft nicht das interne Zwischenablagenformat, das XMLSpy zum Kopieren und Einfügen verwendet.

Tabellen-Ansicht
Weiters können Sie einstellen, wie XMLSpy sich wiederholende Elemente in der Tabellenansicht darstellen soll.

Ansicht

Auf dem Register "Ansicht" können Sie die Darstellung der XML-Dokumente in XMLSpy anpassen.



Enhanced Grid-Ansicht
Reduzierte Elemente in der Enhanced Grid Ansicht können als Vorschau angezeigt werden. Attribute und deren Werte werden in Grau in der gleichen Zeile wie das zugehörige Element dargestellt. Sie können beim Bearbeiten der Dateien automatisch die optimale Breite anwenden und die maximale Höhe und Breite der Spalte festlegen.

Text-Ansicht
Deaktivieren Sie den Zeilenumbruch in der Text-Ansicht.

Programmlogo
Das Programmlogo beim Starten kann deaktiviert werden, um die Applikation schneller starten zu können.

Titelleiste-Inhalt
Im Fenstertitel jedes Dokumentfensters kann entweder nur der Dateiname oder der komplette Pfad angezeigt werden.

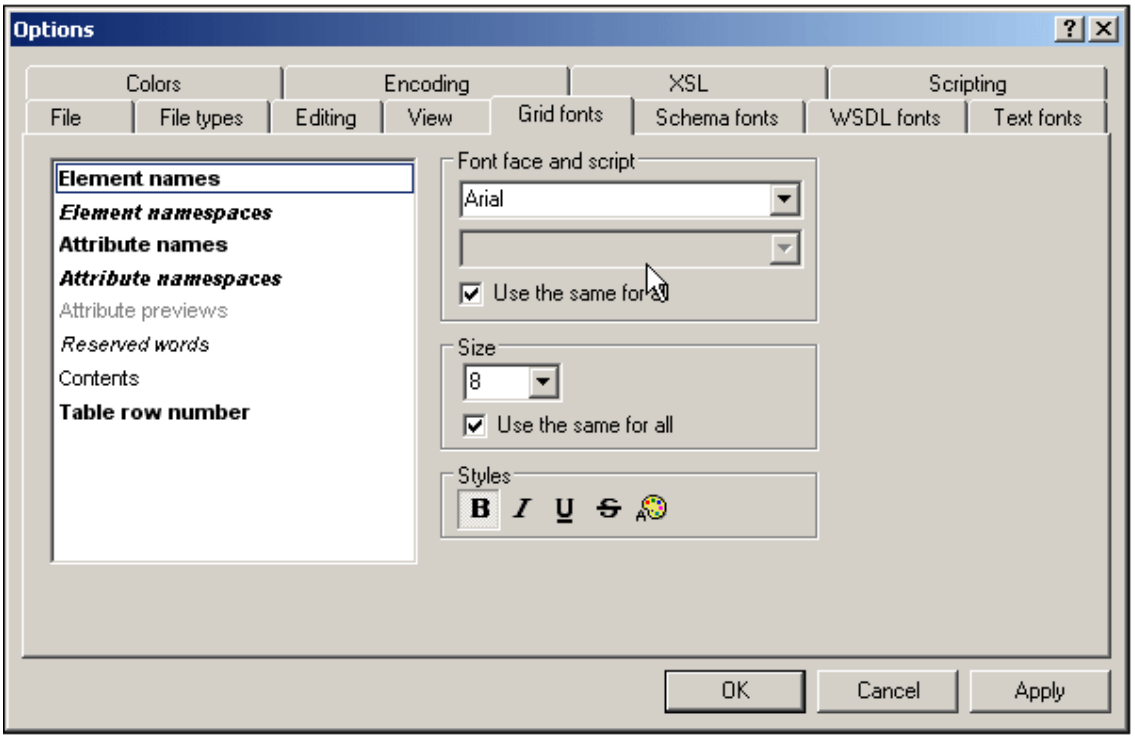
© 2007 Altova GmbH

Authentic-Ansicht
XML-Dateien basierend auf einem **StyleVision Power Stylesheet** (*.sps), werden bei Aktivieren dieser Option automatisch in der Authentic-Ansicht geöffnet.

Browser-Ansicht
Sie können die Browser-Ansicht in einem separaten Fenster öffnen, sodass Sie die Ansicht, in der Sie arbeiten, und die Browser-Ansicht nebeneinander sehen.

Grid-Schriftarten

Auf dem Register **Grid-Schriftarten** können Sie das Aussehen von Texten in der Enhanced Grid-Ansicht anpassen.



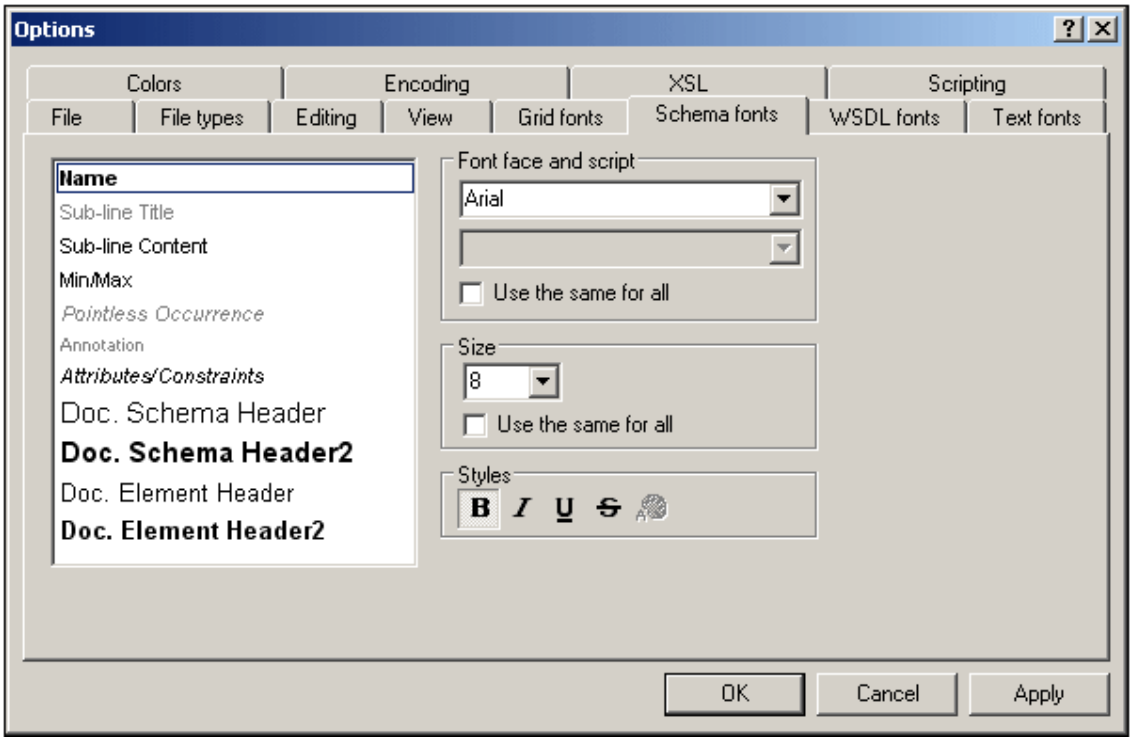
Schriftart und Schrift auswählen
Wählen Sie Schriftart und Schriftgröße, die in der Enhanced Grid Ansicht verwendet werden sollen. Dieselben Einstellungen werden auch für das Drucken verwendet. Deswegen können nur **TrueType** Schriftarten ausgewählt werden.

Größe
Wählen Sie hier die gewünschte Größe aus. Wenn Sie für alle Elemente die selbe Schriftgröße verwenden möchten, aktivieren Sie das Kontrollkästchen "Auf alle anwenden".

Stil
Stil und Farbe können im über die Auswahlschaltflächen eingestellt werden. Die Einstellungen werden sofort als Vorschau in der in Liste auf der linken Seite angezeigt. So sehen Sie sofort, wie Ihr Dokument aussehen wird.

Schemaschriftarten

Auf dem Register **Schemaschriftarten** können Sie das Aussehen von Texten in der Schema/WSDL-Ansicht anpassen.



Schriftart und Schrift auswählen
Wählen Sie Schriftart und Schriftgröße, die in der Schema Design-Ansicht verwendet werden sollen. Dieselbe Schriftgröße und Schriftart wird zum Drucken und Erstellen von Schema Dokumentation verwendet. Deswegen können nur TrueType Schriftarten ausgewählt werden. Komponenten mit dem Präfix "Doc." werden in der Schema-Dokumentation verwendet.

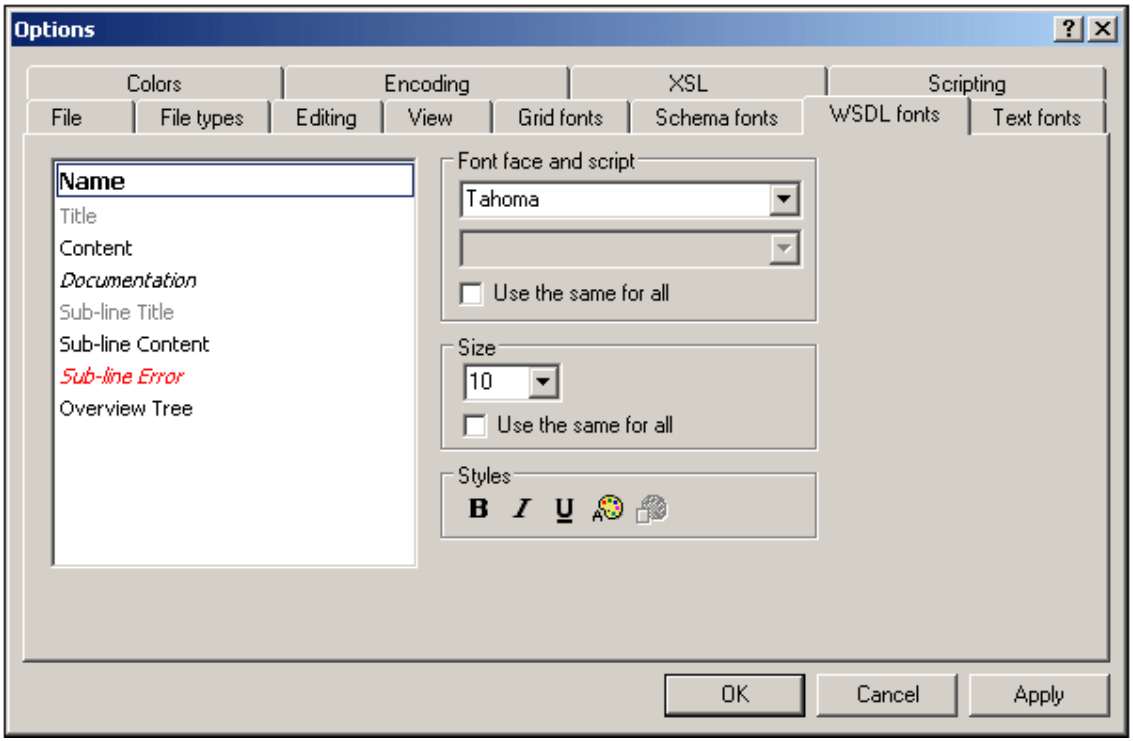
Größe
Wählen Sie hier die gewünschte Größe aus. Wenn Sie für alle Elemente die selbe Schriftgröße verwenden möchten, aktivieren Sie das Kontrollkästchen "Auf alle anwenden".

Stile
Stil und Farbe können über die Auswahlschaltflächen eingestellt werden. Die Einstellungen werden sofort als Vorschau in der in Liste auf der linken Seite angezeigt. So sehen Sie sofort, wie Ihr Dokument aussehen wird.

WSDL-Schriftarten

Auf dem Register **WSDL-Schriftarten** können Sie das Aussehen von Text in der WSDL-Ansicht anpassen.

© 2007 Altova GmbH



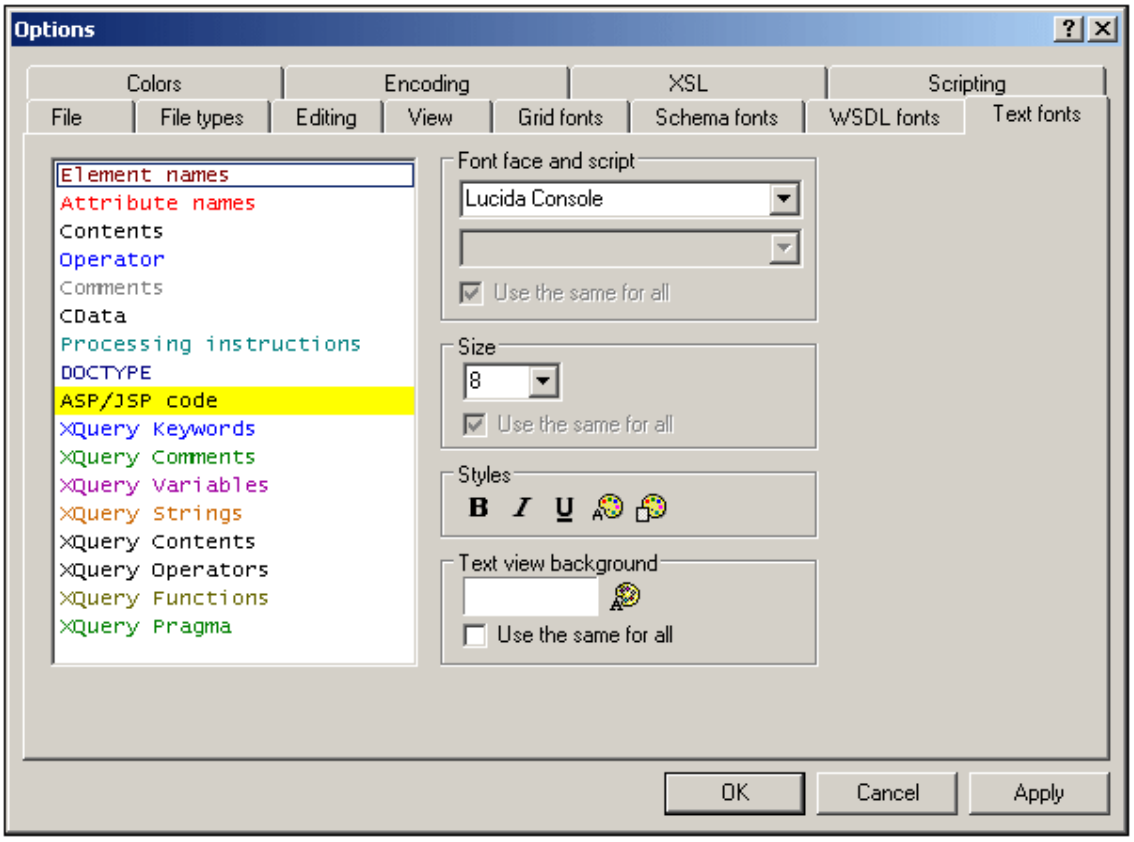
Schriftart und Schrift
Sie können die Schriftart und die Schriftgröße auswählen, in der die verschiedenen Objekte in der WSDL Design-Ansicht dargestellt werden. Dieselben Schriftarten werden auch beim Drucken und Erstellen von Dokumentation verwendet, daher sollten nur True Type-Schriftarten verwendet werden.

Größe
Wählen Sie die gewünschte Größe aus. Wenn Sie dieselbe Größe für alle Einträge verwenden möchten, klicken Sie auf das Kontrollkästchen "Auf alle anwenden"

Stile
Sie können über die Optionen in diesem Bereich Stile und Farben auswählen. Die aktuellen Einstellungen werden sofort in der Liste auf der linken Seite angezeigt, so dass Sie sehen, wie Ihr Dokument aussehen wird.

Textschriftarten

Auf dem Register **Textschriftarten** können Sie das Aussehen von Texten in der Text-Ansicht anpassen.

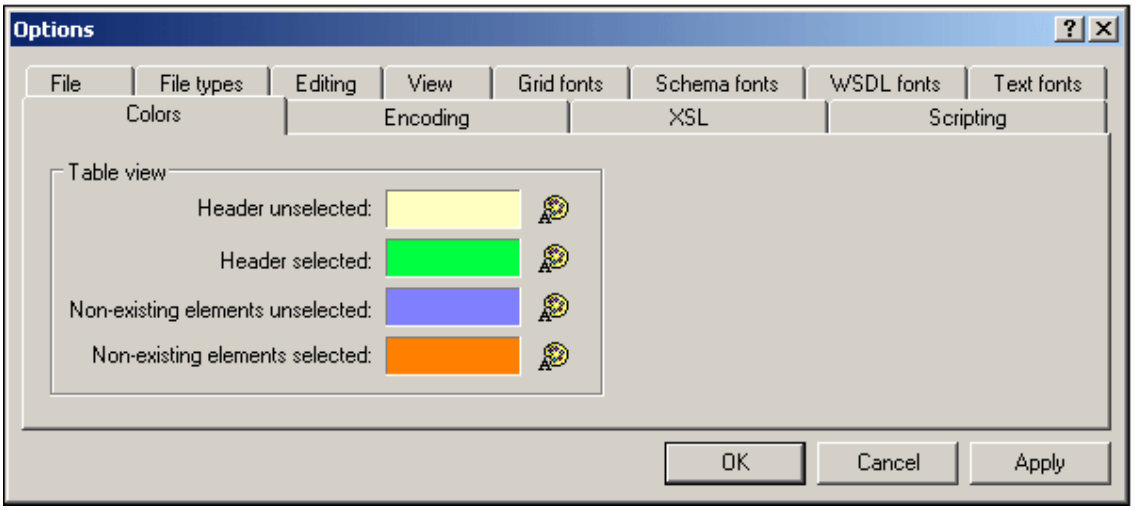


Die im linken Bereich aufgelisteten Typen sind XML-Node-Tpyen, ASP/JSP-Code und XML-Syntax-Komponenten. Sie können die allgemeine Schriftart, den Stil und die Größe des in der Textansicht gezeigten Texts definieren. Beachten Sie, dass für alle Textarten dieselbe Schriftart, derselbe Stil und dieselbe Größe verwendet werden. Sie können für einzelne Textarten nur Textfarbe und Hintergrundfarbe ändern. Dies aktiviert die Syntaxfärbung.

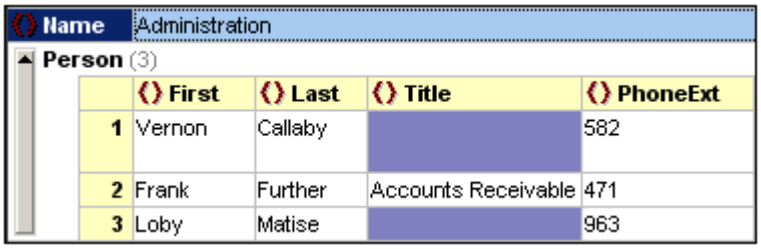
Farben

Auf dem Register **Farben** können Sie die in der Tabellen- oder Grid-Ansicht verwendeten Farben anpassen.

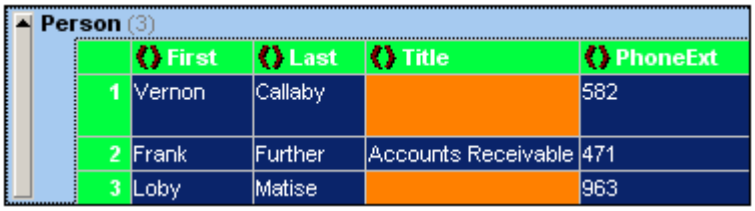
© 2007 Altova GmbH



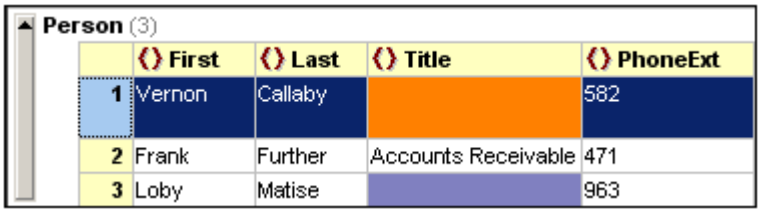
Farben der Tabellenüberschrift
Die Optionen "Tabellenbezeichnung nicht aktiv" und "Tabellenbezeichnung aktiv" beziehen sich auf die Spalten- und Zeilenüberschriften. Im unten gezeigten Screenshot sind die Tabellenbezeichnungen deaktiviert. Die Farbe entspricht den Einstellungen im obigen Dialogfeld.



Die im Feld "Tabellenbezeichnungen aktiv" definierte Farbe wird aktiviert, wenn alle Überschriften ausgewählt sind (*Screenshot unten*) - nicht aber, wenn nur einzelne Überschriften ausgewählt werden. Sie sehen dies im nachfolgenden Screenshot, in dem die im obigen Dialogfeld definierten Farben verwendet werden. Die Auswahl aller Tabellenüberschriften erfolgt durch Klicken auf die Zelle, in der sich beide Überschriftszeilen schneiden oder durch Auswahl des Elements, das als Tabelle dargestellt wird, bzw. durch Auswahl eines seiner übergeordneten Elemente.



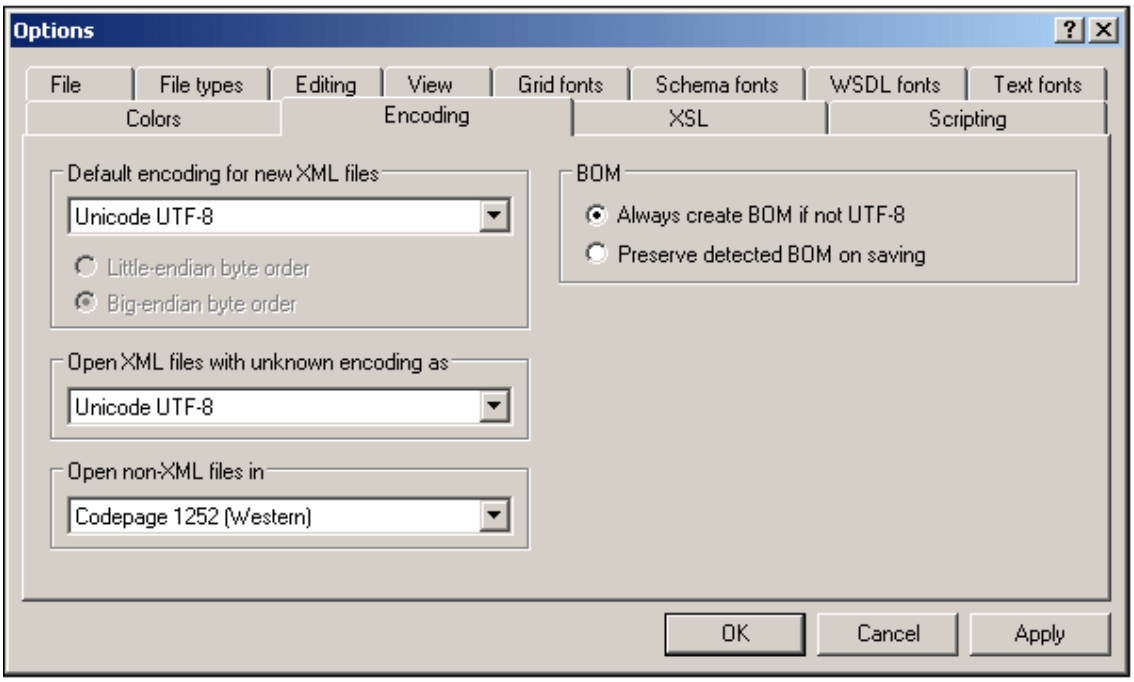
Nicht-existente Elemente
Wenn ein Element oder ein Attribut im XML-Dokument nicht vorhanden ist, kann es unterschiedliche Hintergrundfarben zugewiesen erhalten, wenn es ausgewählt und nicht ausgewählt ist. Ein Beispiel dafür sehen Sie im nachfolgenden Screenshot, in welchem die erste Zeile ausgewählt ist.



Bitte beachten Sie: Zusätzlich zu den hier definierten Farben, verwendet XMLSpy die normalen Einstellungen für Menüfarben und hervorgehobenen Text, die in der Systemsteuerung Ihrer Windowsinstallation unter Anzeigeeinstellungen festgelegt sind.

Kodierung

Auf dem Register **Kodierung** definieren Sie die Einstellungen für die Dateikodierung.



Standardkodierung für neue XML-Dateien
Die Standardkodierung für neue Dateien kann im Dialogfeld "Optionen" durch Auswahl einer Option aus der Dropdown-Liste definiert werden, sodass jedes neue Dokument automatisch mit der entsprechenden XML-Deklaration und der hier gewählten Kodierungsspezifikation erstellt wird. Wird eine 2- oder 4-Byte-Kodierung als standardmäßige Kodierung ausgewählt (z.B. UTF-16, UCS-2 oder UCS-4) können Sie weiters wählen, ob die Byteordnung für XML-Dokumente Little Endian oder Big Endian sein soll.

Die Kodierung für existierende Dateien wird immer beibehalten und kann nur mit dem Befehl **Datei | Kodierung** geändert werden.

Dateien mit unbekannter Kodierung öffnen als
Sie können in dieser Auswahlliste auswählen, mit welcher Kodierung eine Datei geöffnet werden soll, die keine Kodierungsspezifikation enthält oder deren Kodierung nicht automatisch bestimmt werden konnte.

Nicht-XML-Dateien öffnen in

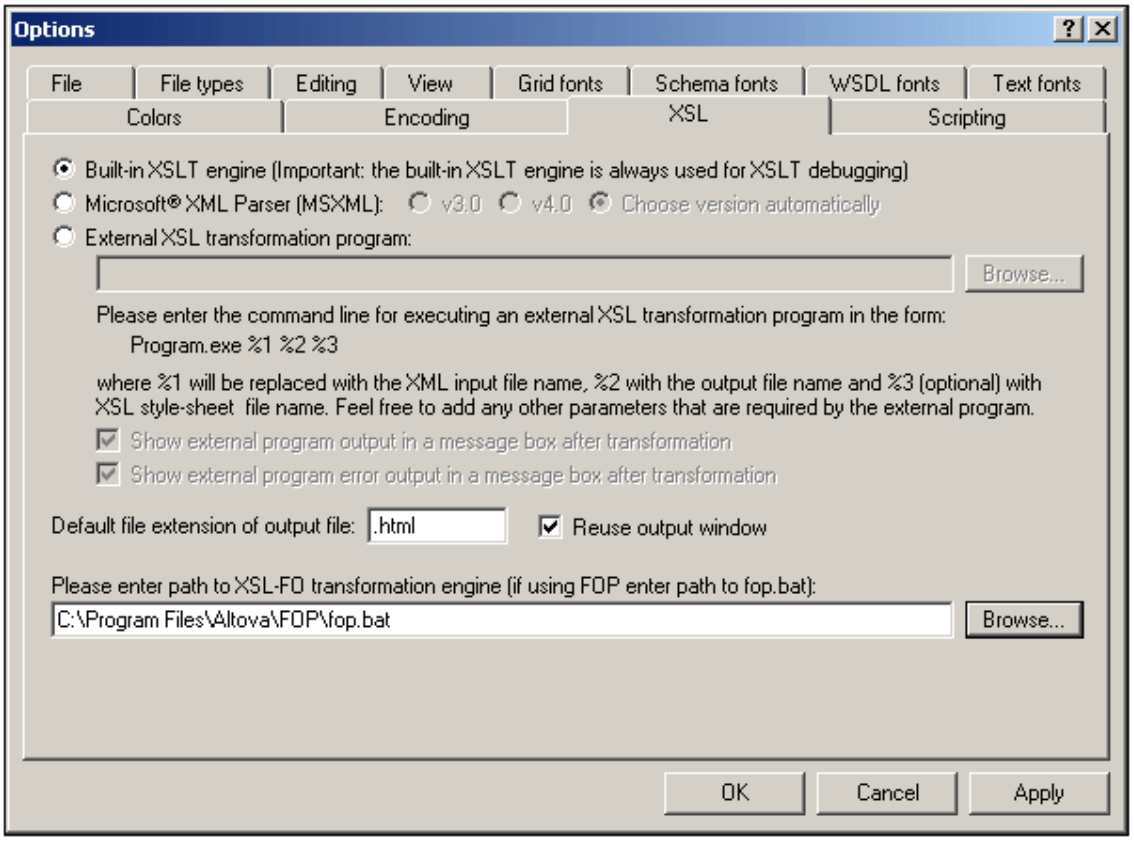
© 2007 Altova GmbH

In dieser Auswahlliste können Sie auswählen, welche Kodierung für nicht XML-Dateien verwendet werden soll. Die können die Kodierung des Dokuments mit dem Befehl **Datei | Kodierung** ändern.

BOM (Byte Order Mark)
Bei Speicherung eines Dokuments mit einer 2-Byte- oder einer 4-Byte-Zeichenkodierung kann es entweder mit (i) der Bytefolge Little Endian und einer Little Endian BOM gespeichert werden (*BOM immer erstellen, wenn nicht UTF-8*); oder (ii) mit der ermittelten Bytefolge und der ermittelten BOM (*Festgestellten BOM beim Speichern beibehalten*).

XSL

Auf dem Register **XSL** werden die Einstellungen für XSLT-Transformationen sowie der Pfad zu einem FO-Prozessor definiert.



XMLSpy enthält den Altova XSLT 1.0 Prozessor und den Altova XSLT 2.0 Prozessor, die Sie für XSLT-Transformationen verwenden können. Je nach dem Wert des `version`-Attributs im Element `xsl:stylesheet` oder `xsl:transform` wird immer der jeweilige XSLT-Prozessor für Transformationen und in Verbindung mit dem XSLT/XQuery Debugger verwendet.

Zur Transformation von XML-Dokumenten kann entweder der MSXML 3.0- oder 4.0 Parser (vorinstalliert), sowie jeder beliebige externe XSLT Prozessor verwendet werden.
Beachten Sie, dass Parameter, die im Dialogfeld "XSLT-Eingabeparameter" eingestellt wurden, nur an die internen Altova XSLT-Prozessoren übergeben werden und nicht an andere XSLT-Prozessoren, die Sie als Standardprozessor für die Verwendung mit XMLSpy definiert haben.

Microsoft XML Parser (MSXML) 3.0 oder 4.0

Wenn Sie Internet Explorer 5 oder höher (z.B. mit dem MSXML Modul) für die Verarbeitung von XSLT verwenden, müssen Sie angeben, ob die resultierende Ausgabe als XML-konformes Dokument (HTML oder WML) oder als nicht-XML konforme Textdatei transformiert werden soll. Die Option "Version automatisch wählen" ist standardmäßig aktiviert.
XMLSpy versucht zuerst Version 4.0 zu verwenden und falls diese nicht verfügbar ist, Version 3.0. Sie können die Version jedoch auch manuell durch Aktivieren des entsprechenden Optionsfelds auswählen.

Externes XSL Transformationsprogramm
Wenn Sie einen externen XSL Prozessor verwenden, müssen Sie angeben, welche Befehlszeile als String verwendet werden soll, damit XMLSpy die XSL-Transformation initiieren kann. In diesem Fall verwenden Sie eine der folgenden Variablen um den String für die Befehlszeile zu erstellen:

%1 XML Instanzdokument, welches verarbeitet werden soll
%2 Ausgabedatei, die erzeugt werden soll
%3 XSL Stylesheet, das verwendet werden soll (optional, wenn das Dokument eine <?xsl-Stylesheet ...?> Referenz enthält)

Der Befehl zum Ausführen einer einfachen Transformation mit dem Saxon (XSLT 1.0) Prozessor lautet z.B.:

```
saxon.exe -o output.xml input.xml stylesheet.xslt
   parameter-name=parameter-value
```

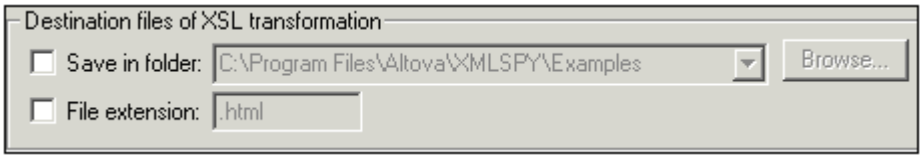
Um XMLSpy für die Arbeit mit dem Saxon-Prozessor zu konfigurieren, aktivieren Sie das Optionsfeld "Externes XSL-Transformationsprogramm" und geben Sie im Textfeld Folgendes ein:

```
c:\saxon\saxon.exe -o %2 %1 %3 parameter-name=parameter-value
```

Weiters können Sie auswählen, ob das Resultat und Fehlermeldungen des externen XSLT Prozessors in XMLSpy in einem Nachrichtenfeld angezeigt werden soll.

Ausgabefenster wiederverwenden:
Ist diese Option deaktiviert, wird sie nur dann angewendet, wenn unter den Ordnereigenschaften auch der Ausgabedateipfad der zugehörigen XML-Datei unter "Speichern in" in einem XMLSpy-Projekt gleichermaßen deaktiviert wurde.

Fenster Ordner-Eigenschaften:

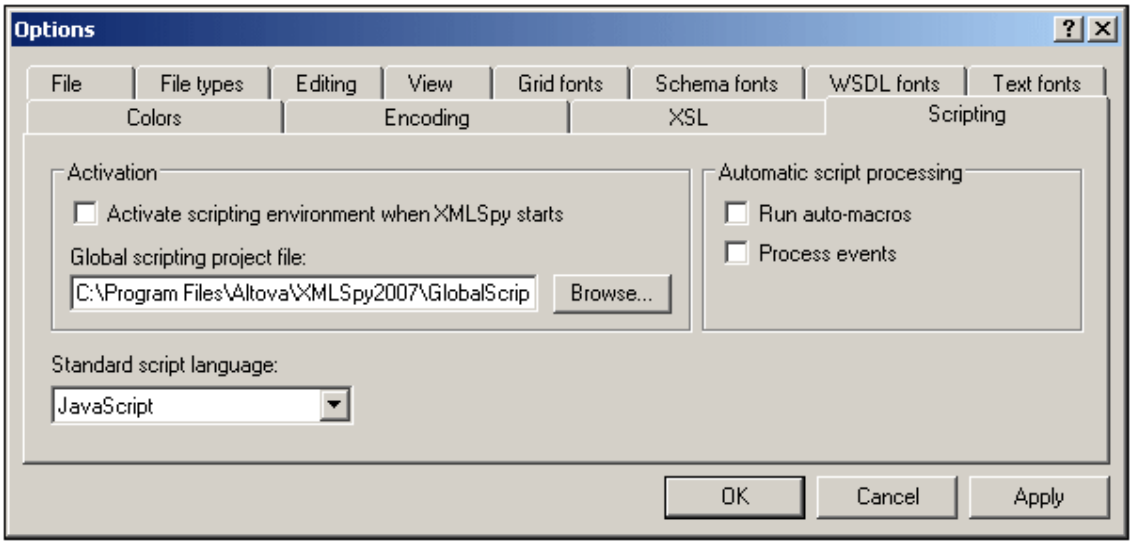


FO-Prozessor
Wenn Sie den Befehl für die FO-Transformation verwenden wollen, müssen Sie den Pfad zur ausführbaren Datei des FO-Prozessors in diesem Feld festlegen.

Skripting-Umgebung

Auf dem Register **Skripting-Umgebung** können Sie die Skripting-Umgebung, sowie weitere spezifische Skriptparameter definieren.

 © 2007 Altova GmbH



Skripting-Umgebung:
Definieren Sie, ob die Skriptumgebung beim Starten von XMLSpy aktiviert werden soll. Klicken Sie auf die Schaltfläche "Durchsuchen...", um die Skripting-Projektdatei auszuwählen.

Standard Skriptsprache:
Wählen Sie die Skriptsprache, die verwendet werden soll: JavaScript oder VBScript.

Automatische Skriptverarbeitung:
In diesem Abschnitt definieren Sie, ob automatische Makros laufen dürfen, und ob die Abarbeitung von Events erlaubt sein soll.

15.16 Menü "Fenster"

Mit Hilfe dieses Menüs können Sie die Dokumentfenster für Ihre XMLSpy Sitzung anordnen. Das Menü "Fenster" enthält Standardbefehle, die aus den meisten Windows-Applikationen bekannt sind.

Sie können die offenen Dokumente hintereinander oder nebeneinander anzeigen. Weiters können Sie Dokumentsymbole neu anordnen (nachdem Sie diese minimiert haben) und die verschiedenen Eingabehilfenfenster ein- oder ausblenden oder direkt über das Menü in ein beliebiges offenes Dokument-Fenster wechseln.

15.16.1 Überlappend

Mit diesem Befehl arrangieren Sie alle offenen Dokumentfenster, sodass sie übereinander und überlappend angezeigt werden.

15.16.2 Horizontal anordnen

Mit diesem Befehl arrangieren Sie alle offenen Dokumente **horizontal nebeneinander**, sodass alle Dokumentfenster gleichzeitig sichtbar sind.

15.16.3 Vertikal anordnen

Mit diesem Befehl arrangieren Sie alle offenen Dokumente **vertikal nebeneinander**, sodass alle Dokumentfenster gleichzeitig sichtbar sind.

15.16.4 Projekt-Fenster

Mit diesem Befehl schalten Sie das Projekt-Fenster ein oder aus.

Dieses Fenster ist abdockbar, d.h. wenn Sie auf die Titelleiste klicken, können Sie das Fenster an jede beliebige Stelle verschieben. Wenn Sie mit der rechten Maustaste auf die Titelleiste klicken, können Sie das Fenster andocken oder ausblenden.

15.16.5 Info-Fenster

Mit diesem Befehl schalten Sie das Info-Fenster ein oder aus.

Dieses Fenster ist andockbar, d.h. wenn Sie auf die Titelleiste klicken, können Sie das Fenster an jede beliebige Stelle verschieben. Mit Rechtsklick docken Sie das Fenster wieder an, oder verbergen es.

15.16.6 Eingabehilfen

Mit diesem Befehl blenden Sie alle drei Eingabehilfe-Fenster ein oder aus.

Dieses Fenster ist andockbar, d.h. wenn Sie auf die Titelleiste klicken können Sie das Fenster an jede beliebige Stelle verschieben. Mit Rechtsklick docken Sie das Fenster wieder an, oder verbergen es.

15.16.7 Ausgabefenster

Bei den Ausgabefenstern handelt es sich um eine Gruppe von drei mit Registern versehenen Ausgabebereichen: (i) Validierungsergebnisse; (ii) Find in Files; und (iii) Ergebnisse der

© 2007 Altova GmbH

XPath-Auswertung. In der Anfangseinstellung werden sie unterhalb des Hauptfensters geöffnet. Mit dem Befehl "Ausgabefenster" können Sie die Ausgabefenster ein- oder ausblenden.

Das Fenster für die Ausgabebereich kann an- oder abgedockt werden. Wenn Sie die Titelleiste ziehen, können Sie es verschieben und als freischwebendes Fenster anordnen. Um das Fenster anzudocken oder auszublenden, klicken Sie mit der rechten Maustaste auf die Titelleiste.

Eine ausführliche Beschreibung der Ausgabefenster finden Sie im Abschnitt "Text-Ansicht" unter Ausgabefenster.

15.16.8 Projekt- und Eingabehilfen

Mit diesem Befehl können Sie das Projektfenster und die Eingabehilfen gleichzeitig ein- oder ausblenden.

15.16.9 Alle ein/aus

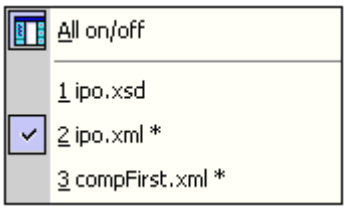
Mit diesem Befehl werden alle andockbaren Fenster zusammen ein- oder ausgeschaltet:

- Projekt-Fenster
- Info-Fenster
- alle drei Eingabehilfen-Fenster

Diese Funktion ist besonders hilfreich, wenn Sie schnell alle anderen Fenster ausblenden möchten, um Ihr Dokument in maximaler Größe anzeigen zu können.

15.16.10 Liste aller offenen Fenster

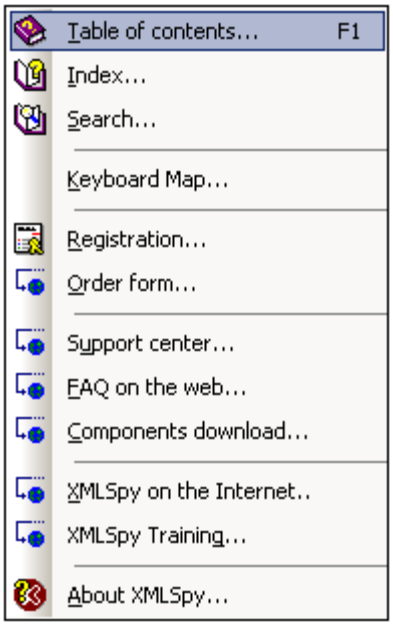
In dieser Liste werden alle derzeit offenen Fenster angezeigt. So können Sie schnell von einem Dokument zum anderen wechseln.



Hierzu können Sie auch die Tastaturkürzel Strg-TAB oder Strg F6 verwenden.

15.17 Menü "Hilfe"

Im Hilfemenü finden Sie alle Befehle, die Sie benötigen, um weitere Informationen zu spezifischen Themen von XMLSpy zu erhalten, sowie Links zu Informationen und Supportseiten auf unserem Webserver.



Im Hilfemenü finden Sie weiters den Registrierungsdialog, über den Sie Ihren Keycode eingeben können, sobald Sie das Produkt erworben haben.

15.17.1 Inhaltsverzeichnis...

Mit diesem Befehl erhalten Sie eine **hierarchische Darstellung** aller Kapitel und Themen, die in der Onlinehilfe enthalten sind. Verwenden Sie diesen Befehl, um direkt von XMLSpy in das Inhaltsverzeichnis zu gelangen.

Sobald das Hilfefenster geöffnet wurde, können Sie zwischen dem Inhaltsverzeichnis, dem Index, und der Suchfunktion wählen. Auf dem Register "Favoriten" können Sie Lesezeichen zu Kapiteln innerhalb der Hilfe setzen.

15.17.2 Index

Mit diesem Befehl gelangen Sie in das **Stichwortverzeichnis** der Onlinehilfe. Sie können auch das Register Index in der Onlinehilfe verwenden.

Der Index bietet eine Liste von Stichwörtern, mit denen Sie durch Doppelklick zu den jeweiligen Abschnitten gelangen, die das Stichwort behandeln. Wenn es mehrere Abschnitte oder Kapitel gibt, in denen das Stichwort vorkommt, werden Sie aufgefordert, aus einer Liste das gewünschte Kapitel auszuwählen.

15.17.3 Suche

Mit dem Suchbefehl können Sie eine **Volltextsuche** in der gesamten Onlinehilfe starten.

- Geben Sie den Suchbegriff im Suchfeld ein und drücken Sie die Eingabetaste.

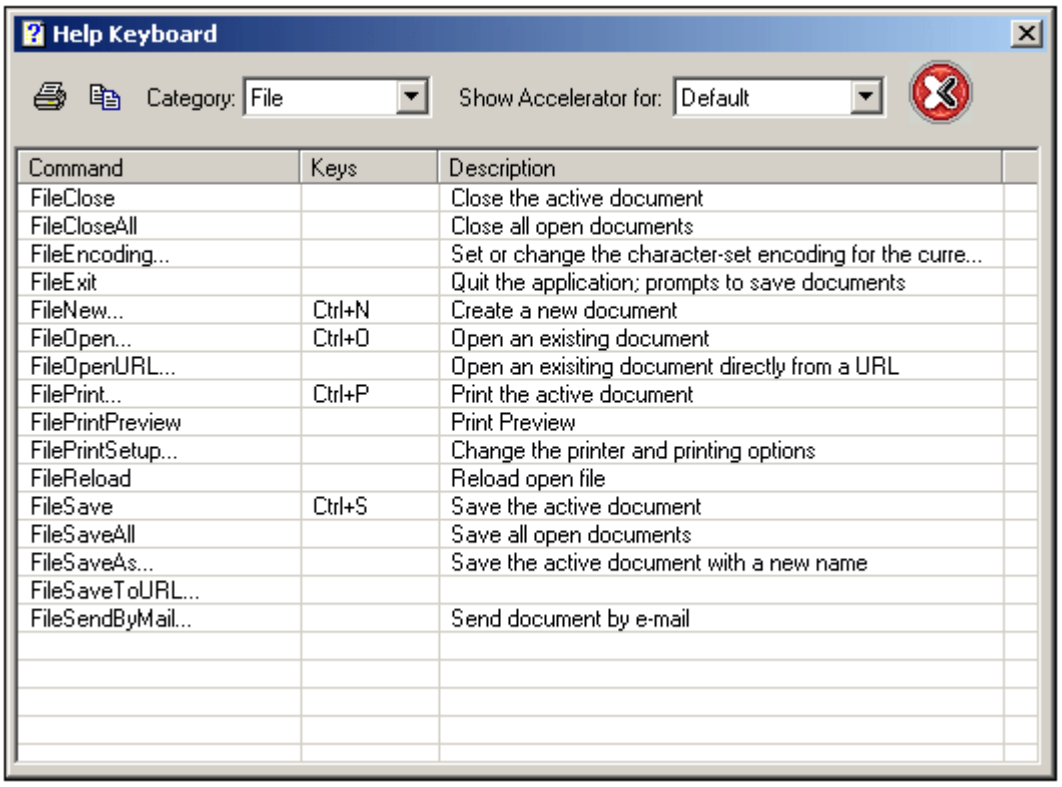
© 2007 Altova GmbH

Die Onlinehilfe zeigt eine Liste von Kapiteln an, in denen der Suchbegriff vorkommt.

- Doppelklicken Sie auf einen Eintrag in der Liste, um das entsprechende Kapitel aufzurufen.

15.17.4 Tastaturzuordnungen

Bei Auswahl des Befehls **Tastenzuordnungen...** wird eine Infobox angezeigt, in der alle Befehle in XMLSpy nach Menüs geordnet angezeigt werden. Die Menübefehle werden mit einer Beschreibung und dem entsprechenden Tastaturkürzel für den jeweiligen Befehl angezeigt.



Um die Befehle eines bestimmten Menüs anzuzeigen, wählen Sie den Menünamen in der Dropdown-Liste "Kategorie" aus. Durch Klicken auf das Druckersymbol können Sie den Befehl ausdrucken.

Beachten Sie bitte die folgenden Anmerkungen zu Tastaturkürzeln:

- Bestimmte Befehle (und Ihre Tastaturkürzel) können nur in einer bestimmten Ansicht angewendet werden. So können z.B. die Befehle im Menü "XML" nur in der Grid-Ansicht angewendet werden. Andere Befehle (wie z.B. **Datei | Speichern** oder **XML | Wohlgeformtheit prüfen**) stehen in mehreren Ansichten zur Verfügung.
- Weitere nützliche Tastaturkürzel: Mit **Umschalt+F10** wird das Kontextmenü in der Textansicht und in der Schema/WSDL-Ansicht aufgerufen; wenn Sie **Strg+E** drücken, während sich der Cursor in der Textansicht innerhalb eines Start- oder End-Tags eines Elements befindet, springt der Cursor zum Ende- bzw. Start-Tag.
- Im Dialogfeld "Anpassen" können Sie auf dem Register "Tastatur" Ihre eigenen Tastaturkürzel für verschiedene Menübefehle definieren.

15.17.5 Aktivierung, Bestellformular, Registrieren, Updates

Software-Aktivierung
Nachdem Sie Ihre Altova-Software heruntergeladen haben, können Sie sie entweder mit Hilfe eines kostenlosen Evaluierungs-Keycode oder eines käuflich erworbenen permanenten Lizenzkeycode aktivieren.

- **Kostenloser Evaluierungs-Keycode.** Wenn Sie die Software zum ersten Mal starten, wird das Dialogfeld "Software-Aktivierung" angezeigt. Es enthält eine Schaltfläche, über die Sie einen kostenlosen Evaluierungs-Keycode anfordern können. Geben Sie in das Dialogfeld, das daraufhin angezeigt wird, Ihren Namen, den Namen Ihrer Firma und Ihre E-Mail-Adresse ein und klicken Sie auf "Jetzt anfordern!". Der Evaluierungs-Keycode wird an die von Ihnen angegebene E-Mail-Adresse gesendet und sollte in wenigen Minuten eintreffen. Geben Sie den Schlüssel nun im Dialogfeld "Software-Aktivierung" in das Feld Key-Code ein und klicken Sie auf OK, um Ihr Altova-Produkt zu aktivieren. Die Software wird für einen Zeitraum von 30 Tagen aktiviert.
- **Permanenter Lizenz Keycode.** Das Dialogfeld "Software-Aktivierung" enthält eine Schaltfläche, über die Sie einen permanenten Lizenz-Keycode erwerben können. Wenn Sie auf diese Schaltfläche klicken, gelangen Sie zum Altova Online Shop, wo Sie einen permanenten Lizenz-Keycode für Ihr Produkt erwerben können. Bei Ihrem permanenten Lizenz-Keycode handelt es sich entweder um eine Einzelbenutzerlizenz oder um eine Mehrfachlizenz. Beide Arten werden per E-Mail zugesandt. Eine *Einzelbenutzerlizenz* enthält Ihre Lizenzdaten, Ihren Namen, die Firma, E-Mail-Adresse und Keycode. Eine *Mehrfachlizenz* enthält Ihre Lizenzdaten, ihren Firmennamen und den Keycode. Beachten Sie, dass gemäß Ihrer Lizenzvereinbarung auf den Computern Ihres Unternehmens nur so viele Versionen der Altova-Software installiert werden dürfen, wie lizenziert wurden (Mehrplatzlizenz)

Anmerkung: Stellen Sie bitte sicher, dass die von Ihnen in das Registrierungsdialogfeld eingegebenen Daten genau mit denen in Ihrer Lizenz-E-Mail übereinstimmen. Geben Sie außerdem bei Mehrplatzlizenzen den Namen des jeweiligen Benutzers in das Feld "Name " ein.

Das Dialogfeld "Software-Aktivierung" kann jederzeit über den Befehl **Hilfe | Software-Aktivierung** aufgerufen werden.

Bestellformular
Sobald Sie eine lizenzierte Version der Software bestellen möchten, klicken Sie im Dialogfeld "Software-Aktivierung" (*siehe oben)* auf die Schaltfläche **Permanenten Key-Code erwerben...** im zum sicheren Online-Shop von Altova weitergeleitet zu werden.

Registrierung
Wenn Sie Ihre Altova-Software nach der Aktivierung das erste Mal starten, erscheint ein Dialogfeld, in dem Sie gefragt werden, ob Sie Ihr Produkt registrieren möchten. Es gibt in diesem Dialogfeld drei Schatflächen:

- **OK:** Leitet Sie zum Registrierungsformular weiter
- **Später erinnern:** Ruft ein Dialogfeld auf, in dem Sie auswählen können, wann Sie das nächste Mal erinnert werden möchten.
- **Abbrechen:** Schließt das Dialogfeld und unterdrückt die Anzeige in Zukunft. Wenn Sie das Produkt zu einem späteren Zeitpunkt registrieren möchten, verwenden Sie den Befehl **Hilfe | Registrieren**

Auf Updates überprüfen
Überprüft, ob am Altova Server eine neuere Version Ihres Produkts vorhanden ist und zeigt

© 2007 Altova GmbH

eine entsprechende Meldung an.

15.17.6 Support Center, Fragen und Antworten, Komponenten downloaden

Support Center
Falls Sie Fragen zu unseren Produkten haben, kontaktieren Sie über diesen Befehl unser Support-Team. Hier finden Sie Links zu unserer Liste von Fragen und Antworten, Supportformulare und die Email-Adressen, um unser Support-Team zu kontaktieren.

Fragen und Antworten im Web
Um Ihnen den bestmöglichen Support bieten zu können, haben wir eine Liste von Fragen und Antworten im Internet (FAQ) zusammengestellt. Diese Liste wird ständig von unseren Support-Mitarbeitern aktualisiert.

Sollten Sie Probleme mit einem unserer Produkte haben, raten wir Ihnen, zuerst die Fragen und Antworten im Internet zu konsultieren. Oft erhalten Sie hierdurch schnelle Hilfe.

Leider bieten wir derzeit noch keinen telefonischen Support. Ihre E-Mails an das Support-Team werden aber in der Regel innerhalb eines Werktages beantwortet.

Wenn Sie uns Anregungen oder Vorschläge zu XMLSpy unterbreiten möchten, verwenden Sie bitte unseren Fragebogen.

Komponenten und Gratistools downloaden
Dieser Befehl ist ein Link zum Komponenten Download Center von Altova im Internet. Von hier können Sie Software-Komponenten verschiedener anderer Anbieter herunterladen, die Sie mit Altova Produkten verwenden können. U.a. finden Sie hier den Altova Validator, XSLT- und XQuery-Prozessoren.

15.17.7 Internet...

XMLSpy im Internet
Mit diesem Befehl gelangen Sie zum Altova Webserver auf http://www.altova.com. Hier erhalten Sie alle Informationen, Produkt-Updates und weitere Angebote von Altova.

Training
Mit diesem Befehl gelangen Sie zur Online-Training-Seite der Altova Website. Hier können Sie sich für Online-Kurse zu unseren Altova-Produkten anmelden.

15.17.8 Über...

Mit diesem Befehl wird der XMLSpy Splashscreen und das Dialogfeld mit urheberrechtlichen Informationen angezeigt.

Bitte beachten Sie:
In diesem Dialogfeld wird die Versionsnummer angezeigt. Der Build sowie die komplette Versionsnummer der Version, die Sie verwenden, werden in der Statusleiste angezeigt.

Altova XMLSpy 2007 Enterprise Edition

Referenz für Programmierer

Referenz für Programmierer

XMLSpy als Automation Server
XMLSpy ist ein Automation Server, d.h. es ist eine Applikation, die anderen Applikationen (den so genannten Automation Clients) programmierbare Objekte zur Verfügung stellt, sodass der Automation Client direkt auf die vom Automation Server verfügbar gemachten Objekte zugreifen kann. Dies ist deshalb von Vorteil für einen Automation Client, da er die Funktionalitäten von XMLSpy nützen kann. So kann ein Automation Client z.B. die XML-Validierungsfunktionen von XMLSpy verwenden. Entwickler können daher ihre Applikationen verbessern, indem Sie die vorhandenen Funktionalitäten von XMLSpy in ihren Applikationen nutzen.

Die programmierbaren Objekte von XMLSpy werden den Automation Clients über die **XMLSpy API**, eine COM API, zur Verfügung gestellt. In dieser Dokumentation finden Sie das Objektmodell der API und eine komplette Beschreibung der verfügbaren Objekte (siehe Abschnitt XMLSpyAPI). Außerdem kann die API auch mit einer Java-Implementierung verwendet werden. Im Kapitel Using the XMLSpy API with Java finden Sie eine Beschreibung dazu.

Anpassen von XMLSpy und Modifizieren der Funktionen
Sie können die Installation von XMLSpy durch Ändern und Hinzufügen von Funktionalitäten anpassen. Außerdem können Sie Formulare für Benutzereingaben erstellen und die Benutzeroberfläche so anpassen, dass sie bestimmte neue Menübefehle und Symbolleistenschaltflächen enthält. All diese Funktionen werden in Form von Skripts implementiert, die mit Objekten der XMLSpy API interagieren. Um diese Funktionen effizient umsetzen zu können, bietet Ihnen XMLSpy eine integrierte Skripting-Umgebung. Im Abschnitt Scripting finden Sie eine komplette Beschreibung der Verwendung der Skripting-Umgebung und der darin verfügbaren Funktionen.

Des Weiteren können Sie XMLSpy mit Hilfe von externen Skripts manipulieren. So könnten Sie z.B. ein Skript schreiben, das XMLSpy zu einem bestimmten Zeitpunkt öffnet, anschließend die XML-Datei in XMLSpy öffnet, die Datei validiert und ausdruckt. Diese externen Skripten würden sich zur Durchführung dieser Aufgaben ebenfalls der XMLSpy API bedienen. Eine Beschreibung der XMLSpy API finden Sie im Abschnitt XMLSpyAPI.

Erstellen von Plug-Ins für XMLSpy
XMLSpy gestattet Ihnen, Ihre eigenen Plug-Ins zu erstellen und in XMLSpy zu integrieren. Sie können dies mit Hilfe der speziellen Plug-In-Schnittstelle von XMLSpy tun. Eine Beschreibung zur Erstellung von Plug-Ins finden Sie im Abschnitt XMLSpy Plug-ins.

Informationen zur Referenz für Programmierer
Die in dieser Referenz für Programmierer enthaltene Dokumentation besteht aus den folgenden Abschnitten:

- Release Notes, in der die Änderungen seit der letzten Version aufgelistet sind
- Scripting: eine Benutzerreferenz für die Skripting-Umgebung in XMLSpy
- XMLSpy Plug-ins: eine Beschreibung zur Erstellung von Plug-Ins für XMLSpy
- XMLSpy API: eine Referenz für die XMLSpy API
- Using XMLSpy API with Java: eine Referenz zur Verwendung der XMLSpy API in Verbindung mit Java
- XMLSpy Integration: eine Referenz für die Integartion von XMLSpy in andere Applikationen

© 2007 Altova GmbH

1 Release Notes

For each release of the XMLSpy API, important changes since the previous release are listed below.

Automation Interface for XMLSpy 2007: type library version 1.6
Changes since the previous release are as follows:

- The Document interface has the method GenerateDTDOrSchemaEx to automatically generate a DTD or schema for the current XML document
- The GenerateSampleXMLDlg interface has the following additional properties: FillAttributesWithSampleData, FillElementsWithSampleData, ContentOfNillableElementsIsNonMandatory, TryToUseNonAbstractTypes, Optimization, SchemaOrDTDAssignment, LocalNameOfRootElement, NamespaceURIOfRootElement, OptionsDialogAction.
- The following properties of the GenerateSampleXMLDlg interface are obsolete and are no longer supported: NonMandatoryElements - obsolete, TakeFirstChoice - obsolete, FillWithSampleData - obsolete.
- The following enumerations are new: SPYAttributeTypeDefinition, SPYSampleXMLGenerationOptimization, SPYSampleXMLGenerationSchemaOrDTDAssignment.

Automation Interface for XMLSpy 2006sp2: type library version 1.5
Changes since XMLSpy API ver 2006:

- The CodeGeneratorDlg interface now has the properties: CompatibilityMode, CPPSettings_GenerateVC6ProjectFile, CPPSettings_GenerateVSProjectFile. These properties generate project files for Visual Studio 2005.

Automation Interface for XMLSpy 2006: type library version 1.5
Important changes since the previous release (XMLSpy API 2004R4) are as follows:

- The Application interface has the following methods: ReloadSettings, RunMacro, ScriptingEnvironment.
- The Document interface has the following additional methods: GenerateSampleXML, SetExternalIsValid, UpdateXMLData.
- The Document interface has the additional event: OnBeforeValidate.
- The XMLData interface has the following additional declarations: CountChildren, CountChildrenKind, HasChildrenKind, GetChild, GetChildKind.
- A new interface GenerateSampleXMLDlg has been added.

Automation Interface for XMLSpy 2004R4: type library version 1.4
Important changes since the previous release (XMLSpy API 2004R3) are as follows:

Authentic View

- AuthenticView, AuthenticRange and AuthenticDataTransfer interfaces replace DocEdit, DocEditSelection and DocEditDataTransfer.
- The AuthenticView and AuthenticRange interfaces now supersede the old DocEdit interface.
- The DocEdit interfaces have been renamed OldAuthentic and are now obsolete.

- The `DocEditSelection` interface has been renamed `AuthenticSelection` and is now obsolete.
- The `DocEditDataTransfer` interface has been renamed `AuthenticDataTransfer`.
- The functionality of now obsolete interfaces are still available. However, usage of the new interfaces is strongly recommended.

Improved events
All events are available as connection point events to allow for easy and flexible integration of external clients using VBA, C++, VBScript, JScript or Perl. New events supporting the document life cycle have been added. All events pass event context information as parameters. Many events support cancellation of the signaled operation. See the section Overview of Events for more details.

Extension of Document interface
- Supports generation of program code (C++, C# and Java) for schema definitions
- Supports generation of MSWord and HTML documentation for schema definitions
- Supports XSL-FO transformation
- Extension of standard properties for documents
- Direct access to data root element of XML documents

Data import and export
- Choice of database for database import.
- Import of hierarchical data, which was previously done using a hand-written SHAPE statement with ImportFromDatabase, now requires a schema file and the use of ImportFromSchema.

Automation Interface for XMLSpy 2004R3 - type library version 1.3
Important changes since XMLSpy API ver 5 rel 4:

Authentic View
- Completely new interfaces `AuthenticView` and `AuthenticRange` to work with Authentic View.

© 2007 Altova GmbH

2 Scripting

XMLSpy provides a **Scripting Environment** within the XMLSpy IDE. The Scripting Environment can be used to:

- Graphically design Forms for user input and for output
- Create, manage, and store scripts for these Forms
- Create scripts for application Event Handlers and for Macros.

Scripts access and interact with objects in the XMLSpy API, which is a COM API. They thus allow you to modify and add functionality to your installation of XMLSpy. Programming languages that can be used in the Scripting Environment are **JavaScript** and **VBScript**.

Forms
In the Scripting Environment, you build a Form graphically using a palette of Form Objects, such as text input fields and buttons. For example, you can create a Form to accept the input of an element name and to then remove all occurrences of that element from the active XML document. For such a Form, a function script can be associated with the text input field so as to take an input variable; and an event handler can be associated with a button to start execution of the functionality, which is available in the XMLSpy API. For details, see Creating a Form.

Global Declarations
In the Scripting Environment, a set of Global Declarations contains the variables and functions that will be used by Forms, Event Handlers, and Macros. The functions make use of the XMLSpy API to access XMLSpy functionality.

Event handling
Using the Scripting Environment, you can easily customize XMLSpy by writing scripts for a variety of available events. You can control events that occur both within Forms (Form events) and within the general XMLSpy interface (application events).

Scripting code for an event is executed immediately upon the triggering of an event. Most events have parameters which provide detailed information on the event. The return value from the script typically instructs the application about how to continue its own processing (for example, the application may not allow editing). For details, see Creating an Event Handler.

Macros
Macros are used to implement complex or repetitive tasks. Macros do not use either parameters or return values. In a Macro, it is possible to access all variables and functions declared in the global declarations and to display forms for user input. Please see Writing a Macro for a simple example of creating a Macro. Also see Calling macros for a description of the ways in which a Macro can be called.

2.1 Overview

This overview of the Scripting Environment discusses the following:

- The relationship between scripting projects and XMLSpy projects
- How and when Forms, Event Handlers and Macros are executed within XMLSpy
- How settings for the Scripting Environment are to be made (in XMLSpy)

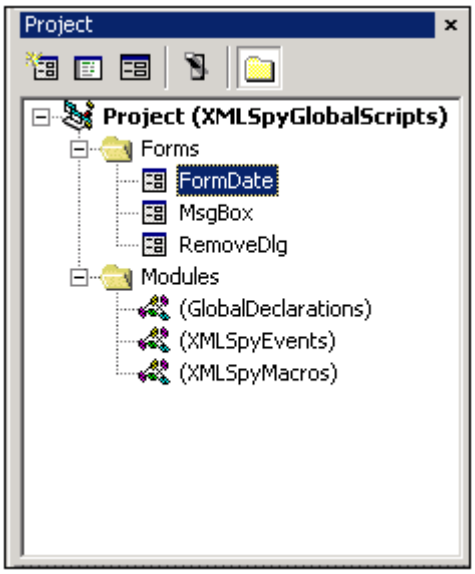
2.1.1 Scripting projects

Working with scripting projects

- In the Scripting Environment, all scripts and scripting information are stored in **scripting projects** (**. `prj` files**).
- After you have created and saved a scripting project, you must assign it to an XMLSpy project. The scripts in the scripting project can then be used when the XMLSpy project to which it has been assigned is opened in XMLSpy.
- Only one scripting project can be assigned to an XMLSpy project at a time.
- You can assign a scripting project to more than one XMLSpy project.
- An XMLSpy project uses the scripts in the global scripting project and the scripts in the scripting project assigned to it.
- If you wish to make scripts applicable to the entire XMLSpy application, you must modify the global scripting project.
- The scripts in the global scripting project apply to the entire XMLSpy application even when no XMLSpy project is open.

The contents of a scripting project and how scripting projects are assigned to and unassigned from XMLSpy projects are described below.

Content of a scripting project
The Scripting Environment screenshot below shows the content of a typical scripting project.



The following points should be noted:

- A scripting project consists of two components: a **Forms** component and a **Modules** component. The Forms component contains all the Forms defined for that scripting project. The Modules component comprises three modules: **Global Declarations**,

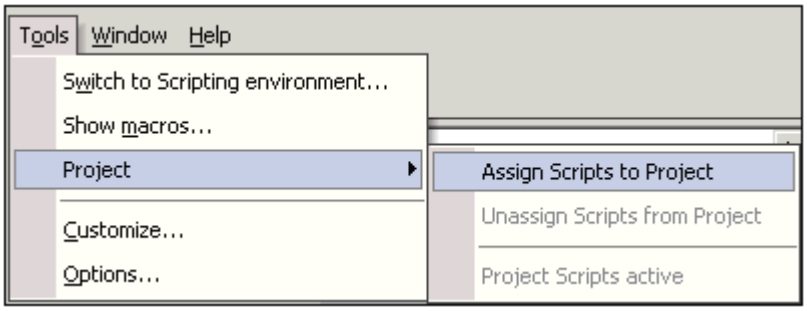
 © 2007 Altova GmbH

XMLSpy Events, and **XMLSpy Macros**.The Global Declarations module contains definitions for variables and functions used in that scripting project. The XMLSpy Events module contains event handler scripts for that scripting project. The XMLSpy Macros module contains the macros that have been defined for that project.

- A scripting project, called `GlobalScripts.prj`, is installed with XMLSpy in the XMLSpy2007 folder. This scripting project contains default global declarations and template scripts for standard XMLSpy event handlers and macro tasks.
- Forms, events, and macros defined in the global scripting project can be accessed from within any XMLSpy project or XML file.
- In addition to the global scripting project, any number of additional scripting projects can be created within the Scripting Environment and saved.

Assigning scripting projects
A scripting project is assigned to an **XMLSpy project** from within the XMLSpy interface with the required XMLSpy project open.

To assign and unassign a scripting project to a project, use the **Tools | Project** submenu (see screenshot below). (These menu items are not enabled if there is no XMLSpy project active currently.)



To assign a scripting project, click **Assign Scripts to Project** and select the scripting project (`.prj` file) from the standard file-selection dialog that appears. The scripting project is loaded, and you are able to edit scripts in it if you switch to the Scripting Environment.

To unassign the current scripting project, select **Unassign Scripts from Project**. The scripting project is unassigned, and the Scripting Environment closes the scripting project.

The following points should be noted:

- For a given XMLSpy project, scripts from the scripting project specific to that XMLSpy project as well as scripts from the global scripting project can be accessed.
- Only **one** scripting project can be assigned to an XMLSpy project at a time.

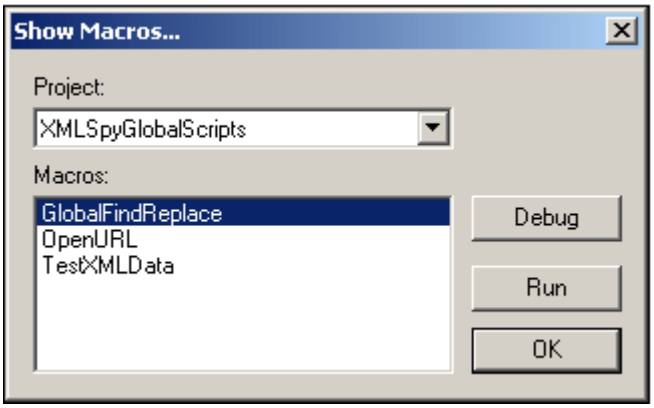
Making a scripting project active
A scripting project is said to be active when it is the active project **in the Scripting Environment**. (More than one scripting project can be open in the Scripting Environment, but only one is active.)

You can activate and deactivate the project-specific scripting project by toggling the **Project Scripts active** command on and off. Note that making the scripting project inactive **does not unassign** it; it merely deactivates the sripting project in the Scripting Environment.

2.1.2 Running Forms, Event Handlers, and Macros

Forms, Event Handlers, and Macros that have been defined in the Scripting Environment are all executed within the XMLSpy interface. However, the way they are called and executed is different for each and has a bearing on how you create your scripting projects.

- A Form is invoked by a call to it either within a function in the Global Declarations module or directly in a Macro.
- An Event Handler runs when the relevant event occurs in XMLSpy. If an Event Handler for a single event is defined in both the global scripting project and the project-specific scripting project, then the Event Handler for the project-specific scripting project is executed first and that for the global scripting project immediately afterwards.
- A Macro is executed from within the XMLSpy interface by clicking **Tools | Show macros...**. This pops up the Show Macros dialog:



In the Project combo box, select either the global scripting project or the scripting project currently assigned to the XMLSpy project (only these two scripting projects are available as options). The macros defined in the scripting project you select appear in the Macros window. Select the macro you wish to run, and click **Run**.

You can also customize your Tools menu to display a list of Macros (**Tools | Show Macros...**). A Macro in this list can subsequently be run by selecting it. See Calling macros for details.

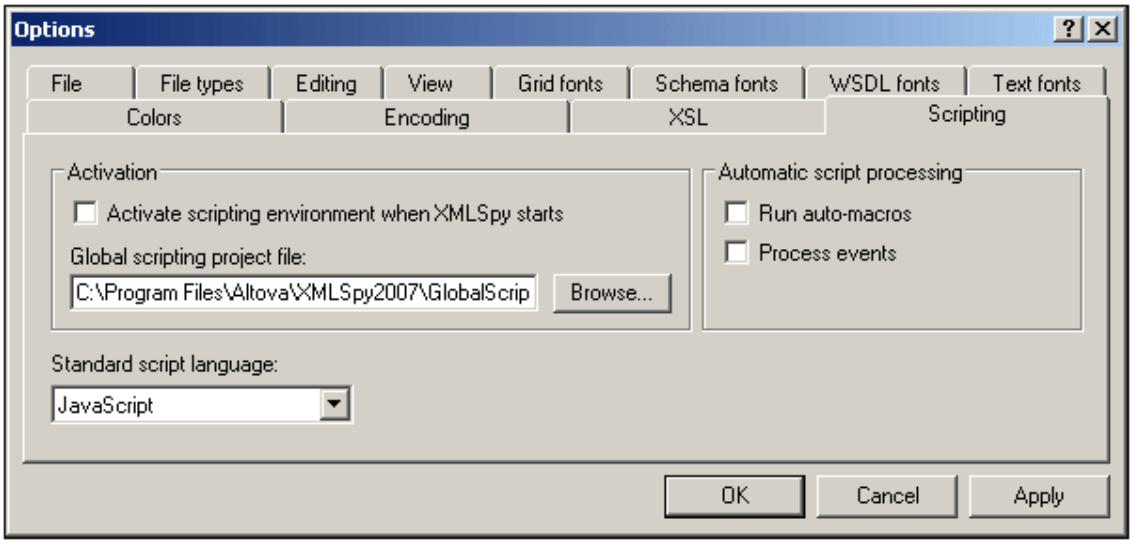
Also see:

- Scripting Settings for information about configuring the XMLSpy interface for scripting.
- Calling macros for details about the various ways Macros can be called.
- The examples of how to create Forms, Global Declarations, Event Handlers, and Macros.

© 2007 Altova GmbH

2.1.3 Scripting Options

The screenshot below shows the Settings dialog for the Scripting Environment of XMLSpy:



Activate scripting environment when XMLSpy starts
If this checkbox is selected, the Scripting Environment is opened as a separate window when XMLSpy starts. Otherwise, the Scripting Environment is not activated until the user selects **Tools | Switch to scripting environment** for the first time.

Global scripting project file
This entry specifies the global scripting project file. A sample global scripting project file is installed with your XMLSpy application. If this file does not exist when the Scripting Environment starts, a new scripting project with the specified name will be created at the specified location.

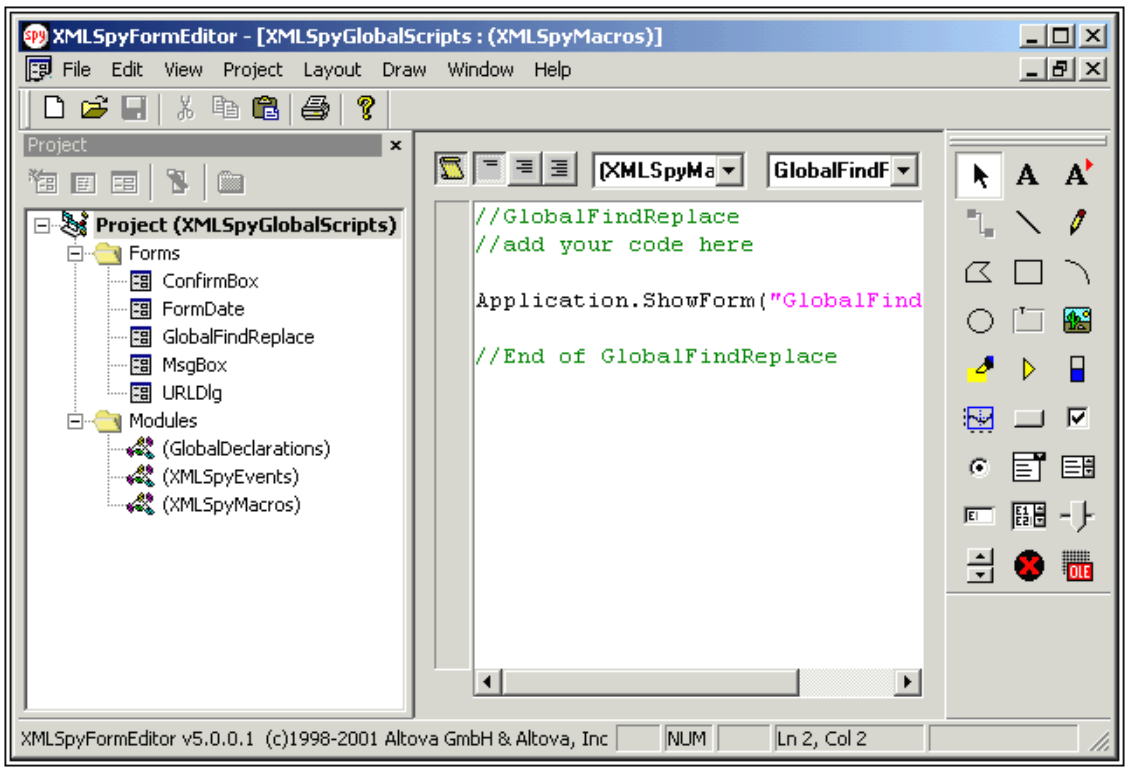
Run auto-macros
Enables or disables the execution of auto-macros. Currently Autorun from the global scripts is the only supported automatic macro. Autorun executes when the scripting environment starts.

Process events
Enables or disables the execution of event handlers. If this check box is not ticked, then Event Handlers in the scripting projects will not be executed.

Standard script language
Every new scripting project created in XMLSpy will propose this scripting language as the default scripting language..

2.2 The Scripting Environment GUI

The Scripting Environment GUI (shown below) is launched from XMLSpy by selecting **Tools | Switch to scripting environment...**. Alternatively, you can specify in XMLSpy (**Tools | Options | Scripting**) that the Scripting Environment be activated automatically each time XMLSpy starts.



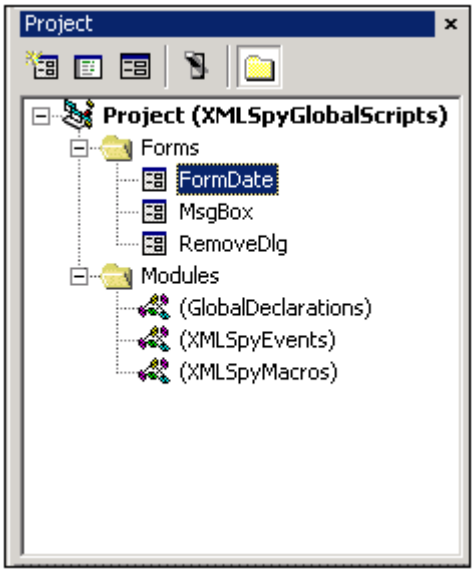
The Scripting Environment window has three parts: a Project Window, Main Window, and Form Object Bar. Each of these parts is described individually below. When a Form is active in the Main Window, the Scripting Environment becomes an effective graphical editor, called the **Form Editor**. The special features of the Form Editor are described separately below.

2.2.1 Project Window

Project Window

The Project Window displays all open scripting projects in a tree view. Only one scripting project can be active at a time. The title of the active project is displayed in bold.

To set a scripting project as the active project, right-click the scripting project and select **Set as active Project**. Closing a project removes it from the Project Window. If a project is not the active project, closing it automatically makes it the active project before it is closed—therefore, after it is closed no project is active.



Double-clicking a Form or Module in the Project Window, opens that Form or Module in the Main Window, where you can graphically design the Form or edit the script.

Icons
The following icons are in the Project Window's title bar:

New Form: Adds a new form to the Forms folder.

View Code: Displays the selected Form or the code of a Module. The code of the selected Form is displayed only if **Layout | Edit Script** has been toggled on for that Form. Otherwise the editable (if **Layout | Edit View** is selected) or non-editable (if neither **Layout | Edit Script** nor **Layout | Edit View** is selected) view of the Form is displayed. This icon is enabled when a Form or Module is selected.

View Form: Displays the selected Form. The editable view of the selected Form is displayed only if **Layout | Edit View** has been toggled on for that Form. Otherwise the code (if **Layout | Edit Script** is selected) or non-editable view (if neither **Layout | Edit View** nor **Layout | Edit Script** is selected) of the Form is displayed. This icon is enabled when a Form is selected.

Run Form: This icon is enabled when a Form is selecteds.

Toggle Folders: Toggles the Forms and Modules level of the tree on and off.

Context menus
Two context-sensitive popup menus are available within the Project Window. The first appears when you right-click the title of a project. The second appears when you right-click a Form or Module. These popup menus give you access to commands available in the **File** and **Project** menus.

2.2.2 Main Window

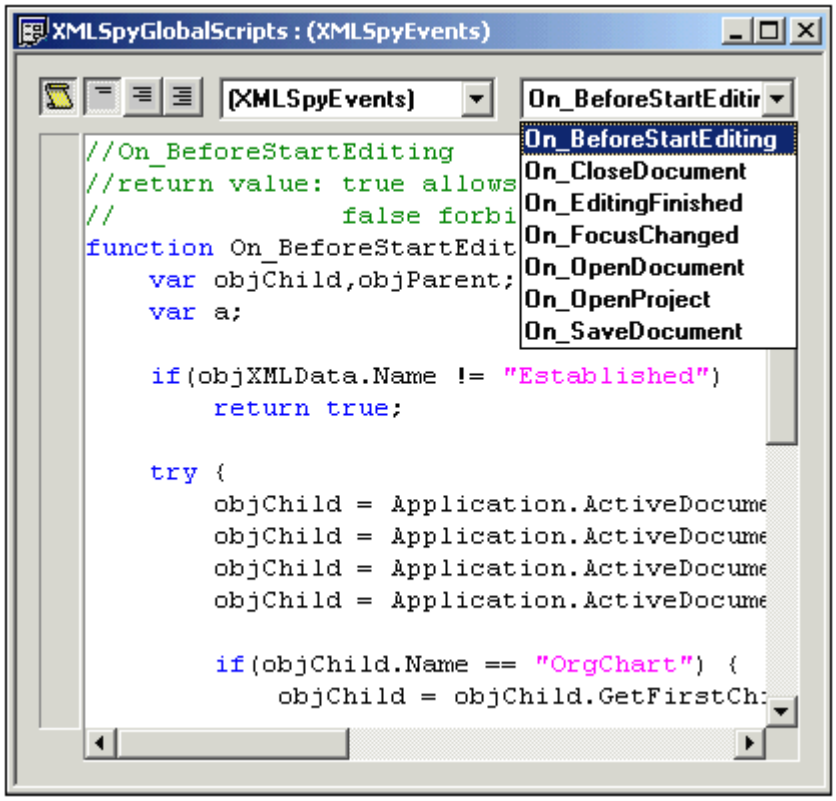
The Main Window is the area in the GUI where individual Forms and Modules are displayed in separate windows. It is in these windows that you design Forms and edit scripts.

Forms

- Each Form is displayed in a separate window.
- When the Scripting Environment GUI has a Form window active, it is called a Form Editor. For details of usage, see The Form Editor.
- A Form is displayed as the graphical dialog box presented to the user (Design View). If **Layout | Edit View** is toggled on for a given Form, then that Form is editable. Otherwise the Design View is non-editable.
- Additionally, if **Layout | Edit Script** has been toggled on for a Form, then the script underlying that Form is displayed in the Form window. Clicking the Close Script View icon switches back to the Design View (editable or non-editable according to whether **Layout | Edit View** is toggled on or off, respectively).
- Note that the **Layout | Edit View** and **Layout | Edit Script** switches are specified separately for each Form.

Modules

- Each Module (Global Declarations, (Application) Events, and Macros) opens in a separate window.
- Within a Module window, the combo box on the left indicates which module is selected (in the screenshot below, the combo box shows that the Events Module is the selected Module). To select a specific item (Global Declaration, Event-Handler, or Macro), click the required option in the drop-down menu of the combo box on the right.



Context menus

Right-clicking within the Script View of a Form or within a Module window pops up a context

 © 2007 Altova GmbH

menu that gives you access to features that help you with your scripting.

2.2.3 Form Object Bar

The Form Object Bar is a tool palette that helps you add controls to your Forms. Select a Form Object from the bar, then place the pointer at the desired location in your Form and draw a rectangle to specify the size of the object.

The most commonly used objects are described below:

Static Text: Adds text fields such as captions or field descriptions.

Button: Adds a button. It is possible to assign bitmaps to these buttons.

Check Box: Adds a check box, which enables Yes–No type elements.

Combo Box: Adds a combo box, which allows the user to select an option from a drop-down menu.

List Box: Adds a list box, which displays a list of items for selection.

Edit Box: Defines a single line of text.

Multi-edit Box: Defines multiple lines of text.

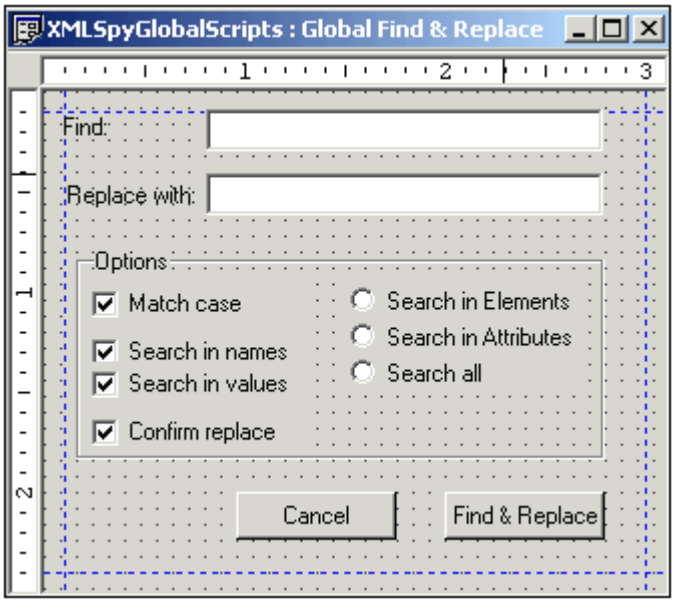
ActiveX: Allows the integration of an ActiveX control in the Form.

Setting Form Object properties and events
Each Form Object has its own set of properties and events. You can specify the settings of a Form Object in the Property Sheet for that object, as well as add some scripting code to the provided events. For details about how to do this, see The Form Editor.

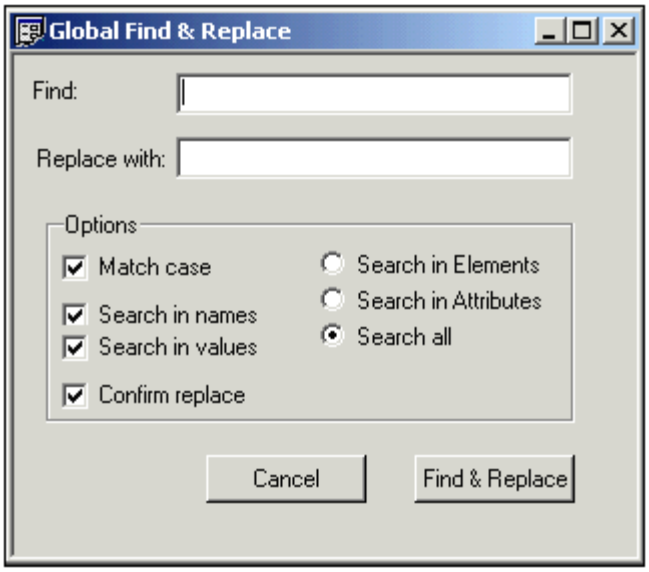
2.2.4 The Form Editor

The Form Editor is the Scripting Environment used for editing a Form. A Form can be opened by double-clicking that Form's icon in the Project Window or by right-clicking a Form icon and selecting either **View Code** or **View Object**.

Form views
Each Form opens in a separate window (shown below) that can be resized and minimized. The two views in which a Form is displayed are Design View and Script View.



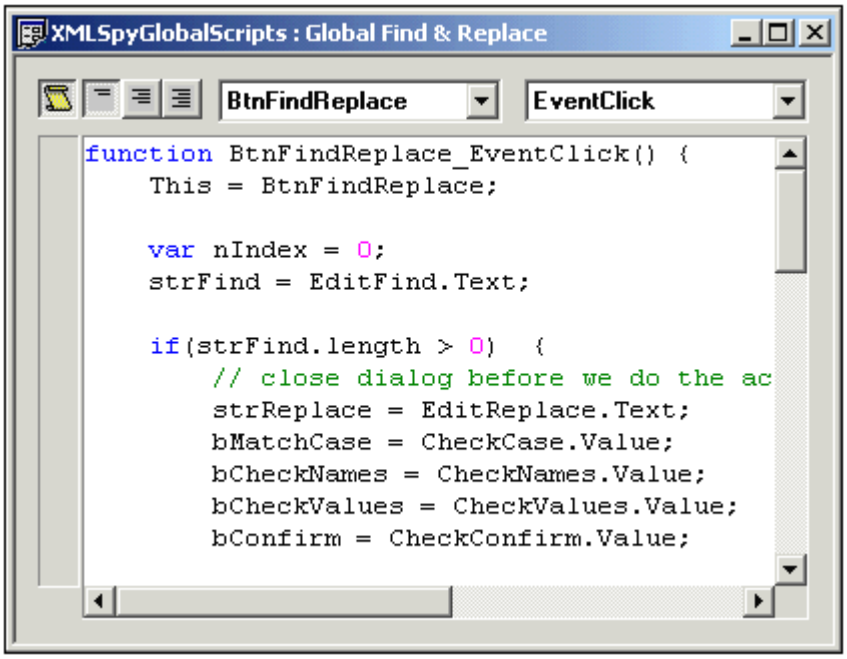
The Design View shown above is editable. In order to make a Design View editable, you must toggle on the menu item **Layout | Edit View**. If this menu item is toggled off, then the Design View is uneditable and the Form appears as it will to the user (see screenshot below).



Additionally, a Form can be displayed in a Script View (shown below) by toggling on the menu item **Layout | Edit Script**. Closing the Script View switches the view back to Design View

© 2007 Altova GmbH

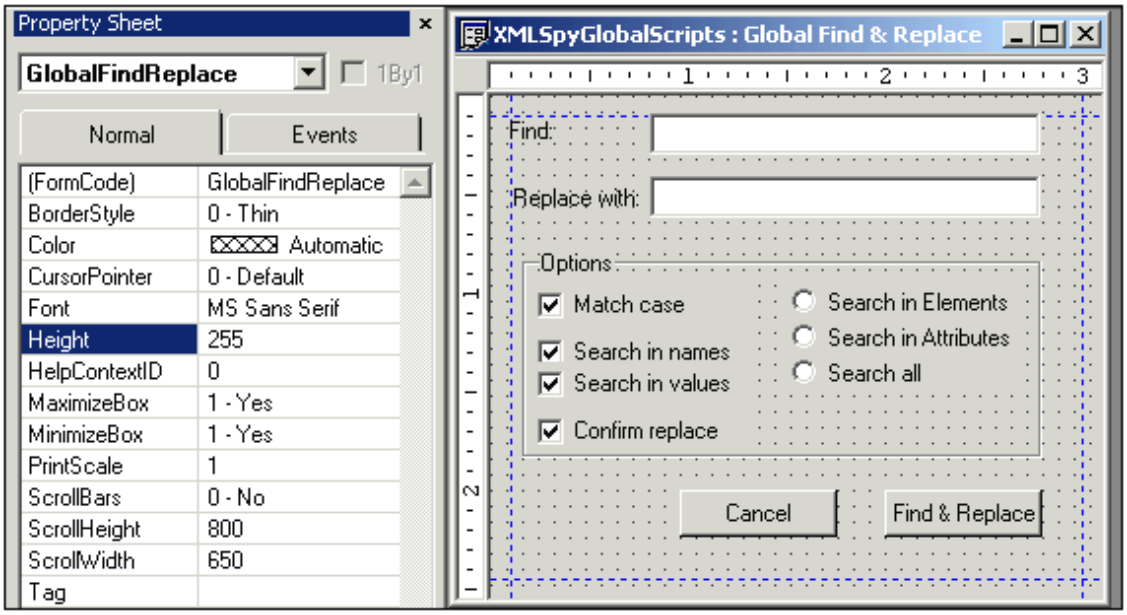
(editable or non-editable according to whether **Layout | Edit View** is toggled on or off, respectively).



Design View
A Form must be created and edited in the editable Design View of the Form. To specify the settings of the workarea, use the menu item **Layout | Grid Settings...**.

In order to insert objects into the Form, use the Form Objects in the Form Object Bar. Build up the visual design of your Form by placing objects from the Form Object Bar at suitable locations within the Form outline. Specify visual refinements of the layout through the **Layout** menu items.

To specify the properties of the Form, either right-click in an empty area of the Form or select **Layout | Properties...**. This displays the Properties Sheet of the Form, where you can define the size of the Form, its font, buttons, and so on.



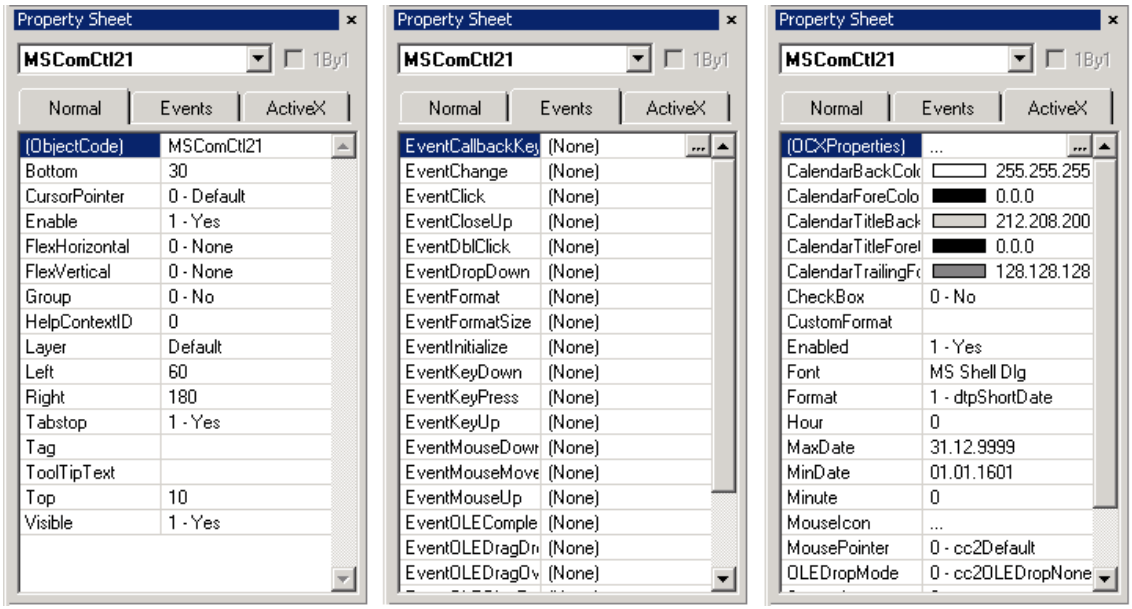
To switch to the Property Sheet of a particular object, either select that object in the design view or select the name of that object from the drop-down list of the combo box in the Property Sheet.

To close the Property Sheet, click the **Close** button.

The Property Sheet
The Property Sheet lists all properties and events of the selected Form or Form Object when the editable Design View of the Form is active. It enables you to modify properties and to add or modify code for properties and Form event handlers. The combo box at the top of the Property Sheet lists all the properties and events of the active Form.

The Property Sheet has three tabs: Normal, Events, and ActiveX. The screenshots below show the three tabs of the Microsoft DatePicker ActiveX control. Note that different Form Objects have different properties and events, and not all objects have three tabs in the Property Sheet.

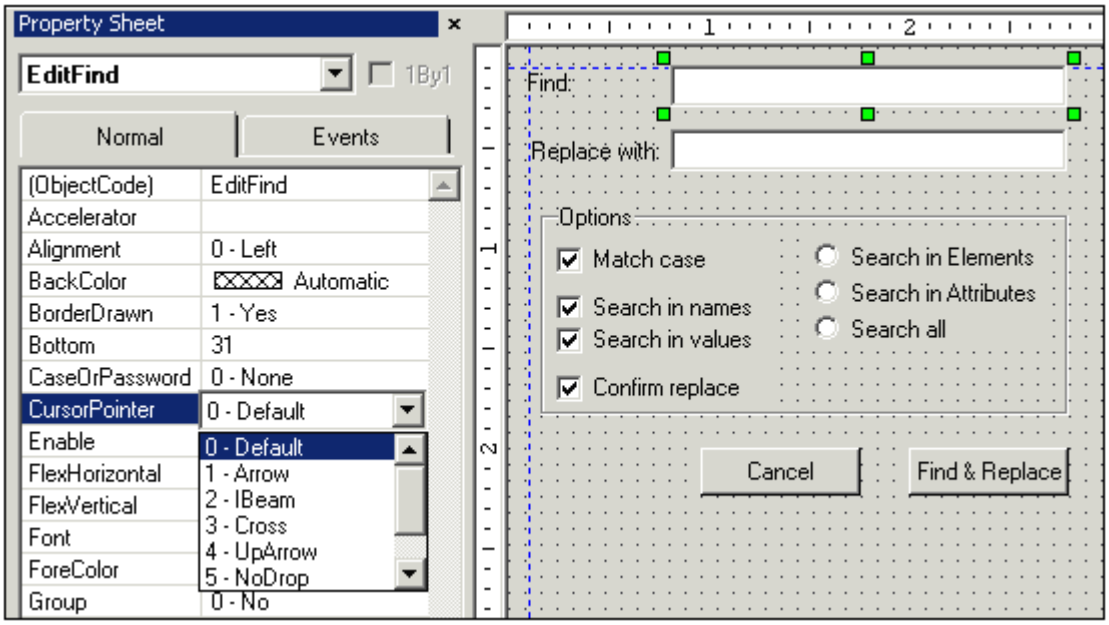
© 2007 Altova GmbH



The **Normal tab** specifies the name of the object and properties relating to the object's appearance in the Form. The name is specified by the `ObjectCode` property (here `MScomCtl21`). If you write a script and need access to methods or properties of this object, you must use this name. For example:

```
var strTmp;
strTmp = "";
strTmp = MSComCtl21.Year + '-' + MSComCtl21.Month + '-' +
   MSComCtl21.Day;
```

If a property has enumerated values, then these become available in a drop-down menu when the property is selected (see screenshot below).



In the above screenshot, the `EditFind` object is selected. When the `CursorPointer` property is selected, a drop-down menu with the possible values for this property becomes available. This enables you to quickly and correctly select a suitable value for the property.

In the **Events and ActiveX tabs**, to enter code for a property, first select the property for which you want to enter code. A button with an ellipsis (. . .) appears at the right of the entry field. Click this button to open a Script Editor window for that code fragment.

Using layers in Form design
When you insert Form Objects while in the Design View of the Form, these objects are created by default on a layer titled "Default". You can add additional layers **below the Default layer** if you wish to create a Form with backgrounds objects. The objects in each layer are superimposed on the objects of underlying layers if there is an overlap. Layers below the Default layer can be moved relative to each other (i.e. up or down), but the Default layer remains the top layer. As a result, objects in the Default layer will be superimposed on all objects in lower layers. For details of using layers, see Layout.

Setting the tab sequence for data-input objects
When users input data in a Form, they commonly move from one input field to the next by pressing the Tab key. You can specify the tab sequence in the Objects Sheet (Tab Order) dialog. See Layout for details.

Also see:

- The Layout menu for commands used to layout a Form. There are a range of commands that help you to format
- Creating a Form for more details about creating a Form.

© 2007 Altova GmbH

2.3 Using the Scripting Environment

The Forms, Event Handlers, and Macros you create in the Scripting Environment use scripts that interact with the XMLSpy API, which is a COM API. The variables and functions required for this interaction are defined in the **Global Declarations** module of the Scripting Environment. The object model of the XMLSpy API is documented in the XMLSpyAPI documentation.

Event Handler scripts can be written for standard XMLSpy events as well as for custom events. This allows you to add and modify event handling in your XMLSpy interface. See Creating an Event-handler for an example of how to work with Event Handlers in XMLSpy.

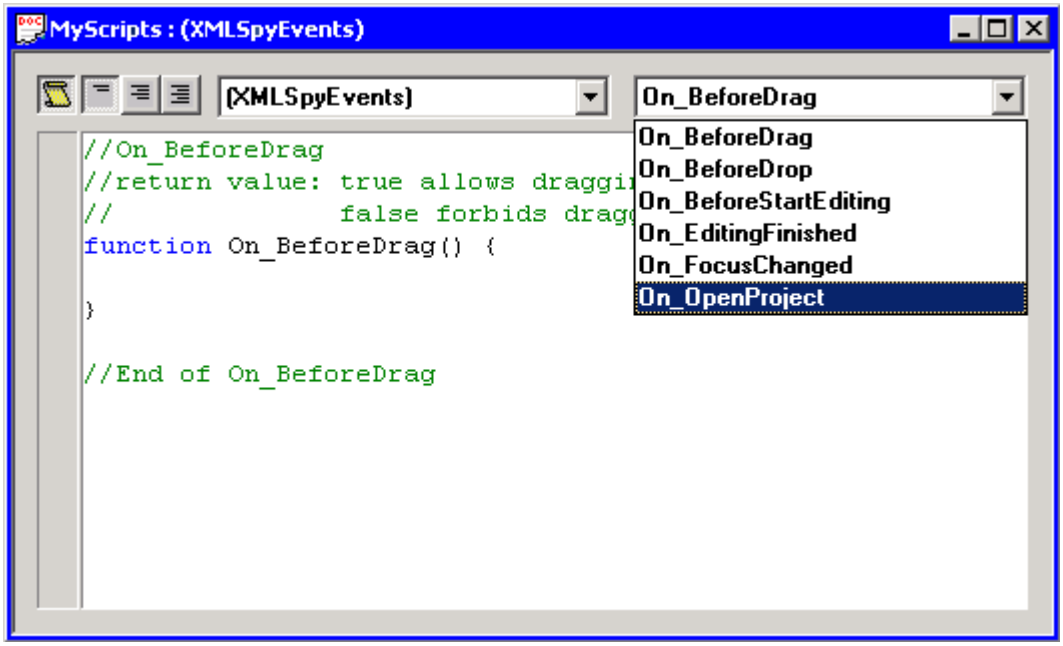
When you create a **Form**, you require variables to accept user input and functions that carry out the required functionality. Both variables and functions are defined in the Global Declarations. In addition, you need to reference some event handler to execute the Form in the desired way. Creating a Form describes the steps required to create a Form.

Macros automate repetitive or complex tasks. In the Scripting Environment, you can create a script that calls functions both XMLSpy functions as well as custom functions you define in the Global Declarations module. This flexibility provides you with powerful method of automating tasks within the XMLSpy IDE. How you create a Macro is described in Writing a Macro.

The subsections of this section explain, with examples, how each of the above is to be created. The section ends with a subsection explaining how Macros are called in XMLSpy.

2.3.1 Creating an Event Handler

To create an Event Handler for application events (as opposed to Form events), double-click the XMLSpyEvents Module in the Project Window. This pops up a scripting window for event handlers in the Main Window.



The combo box on the right lists all the application events. To access a particular event-handler script, click the relevant event in the righ-hand combo box. This pops up the script which you can edit or a script outline you can fill in. If the event you want to write code for does not have an entry in the Event combo box, use the **Project | Add Function** command (or right-click in the script window and select **Add Function**) to add an Event.

Note:

- Event Handlers need function headers with the correct spelling of the event name. Otherwise the Event Handler will not get called.
- It is possible to define local variables and helper functions within macros and event handlers. Example:

```
//return value: true allows editing
//return value: false disallows editing
var txtLocal;
function Helper()
{
  txtMessage = txtLocal;
  Application.ShowForm("MsgBox");
}
function On_BeforeStartEditing(objXMLData)
{
  txtLocal = "On_BeforeStartEditing()";
  Helper();
}
```

- While a Macro is executed, Event Handlers from XMLSpy are not processed.
- The best way to re-create a deleted event handler is to create a new scripting project and copy name and function definition from there.
- In order for events to be processed, the Process Events options must be toggled on in the Settings dialog of XMLSpy. See Scripting Settings for details.
- Also see Programming points to note.

Example
The screenshot above shows the XMLSpyEvents module with a list of predefined events. The `On_OpenProject` allows you to add a script that will be executed each time XMLSpy opens a project. The example script below sequentially opens all XML files located in the XML folder of the project and validates them. If the validation fails the script shows the validation error and stops. If a file passes the validity test, it will be closed and the next file will be opened.

Enter the following script for the `On_OpenProject()` event, and then save the scripting project.

```
function On_OpenProject()
{
 var bOK;
 var nIndex, nCount;
 var objItems, objXMLFolder = null;

 objItems = Application.CurrentProject.RootItems;
 nCount = objItems.Count;

 // search for XML folder
 for(nIndex = 0; nIndex < nCount; nIndex++)  {
  var txtExtensions;
  txtExtensions = objItems.Item(nIndex).FileExtensions;

 if(txtExtensions.indexOf("xml") >= 0)     {
  objXMLFolder = objItems.Item(nIndex);
 break;
 }
}

// does XML folder exist?
if(objXMLFolder) {
 var objChild, objDoc;

 nCount = objXMLFolder.ChildItems.Count;
```


© 2007 Altova GmbH

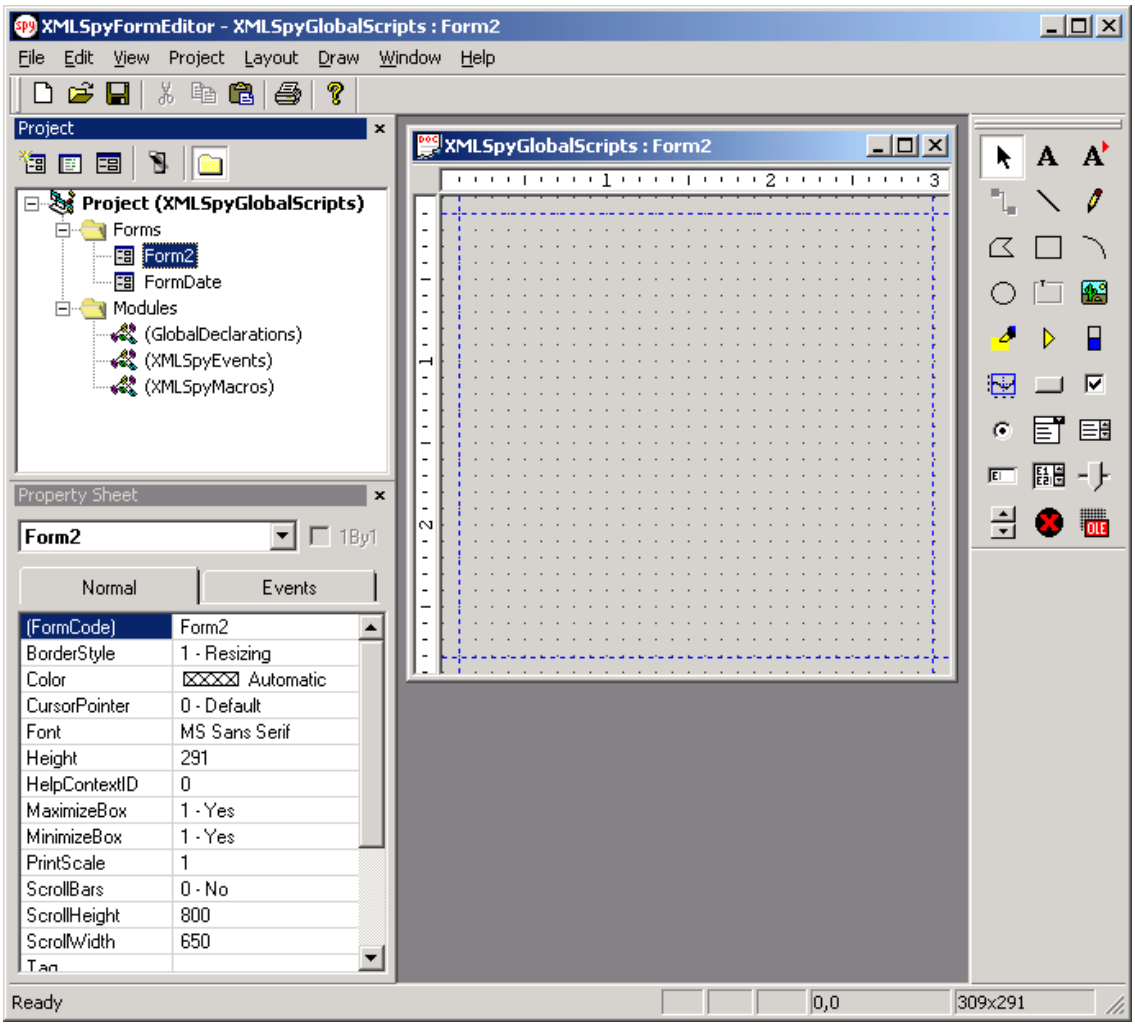
```
// step through associated xml files
for( nIndex = 1;nIndex <= nCount;nIndex++) {
 objChild = objXMLFolder.ChildItems.Item( nIndex);

 try{
  objDoc = objChild.Open();

  // use JScript method to access out-parameters
  var strError = new Array(1);
  var nErrorPos = new Array(1);
  var objBadData = new Array(1);

  bOK = objDoc.IsValid( strError,nErrorPos,objBadData);

  if(!bOK) {
   // if the validation fails, we should display the
   // message from XMLSpy
   // of course we have to create the form "MsgBox" and
   // define the global txtMessage variable
   //
   // txtMessage = Position:" + nErrorPos[0] + "\n" + // strError[0];
   // txtMessage += "\n\nXML:\n" + objBadData[0].Name + ", " +
   //   objBadData[0].TextValue;
   //
   // Application.ShowForm("MsgBox");

    break;
  }

  objDoc.Close( true);
  objDoc = null;
 }
 catch( Err) {
  // displaying the error description here is a good idea

  // txtMessage = Err.Description;
  // Application.ShowForm("MsgBox");

  break;
  }
 }
}
}
```

Testing the Event Handler
Switch to XMLSpy, and open a project to see how the Open Project event is handled.

2.3.2 Creating a Form

A Form queries users for input (data and/or a choice), and executes the required action based on the input. In the Scripting Environment, you can graphically design a Form as well as create the scripts to perform the required action. The example below takes you through the steps needed to create a Form. In it we create a Form in which the user can type in the name of elements that are to be deleted from the active XML file in XMLSpy.

Creating a new Form
To add a new Form to a scripting project, select **Project | Add Form** or click the **New Form** button in the toolbar of the Project Window. The new Form is added immediately. It appears as a separate window (see screenshot below), and its title appears in the Project Window. Open the Property Sheet for the Form by either right-clicking within the Form Window or selecting **Layout | Properties...**.



Naming the Form
You name a Form by entering a name for it in the line `FormCode` of the Property Sheet . For our example, enter `RemoveDlg` as the name of the Form. After entering the name, press Enter. The name of the Form now also appears in the title box of the Propery ySheet.

Inserting a Form Object
We need an edit box in which the user can type the names of the elements to be removed from the XML file. Select the edit box icon from the Form Objects Bar on the right side of the Main Window. Then, in the Design View of the `RemoveDlg` Form, draw, with the mouse, a rectangle for the edit box. This rectangle sets out the position and dimensions of the edit box. The edit box is created in the Design View, and (if **Layout | Properties...** is toggled on) the Property Sheet for the edit box is displayed. (To display the Property Sheet for the edit box, either select the edit box and toggle on **Layout | Properties...** or right-click the edit box.)

In the `( ObjectCode)` line of the Property Sheet, type in `EditElements` as the name of the edit box.

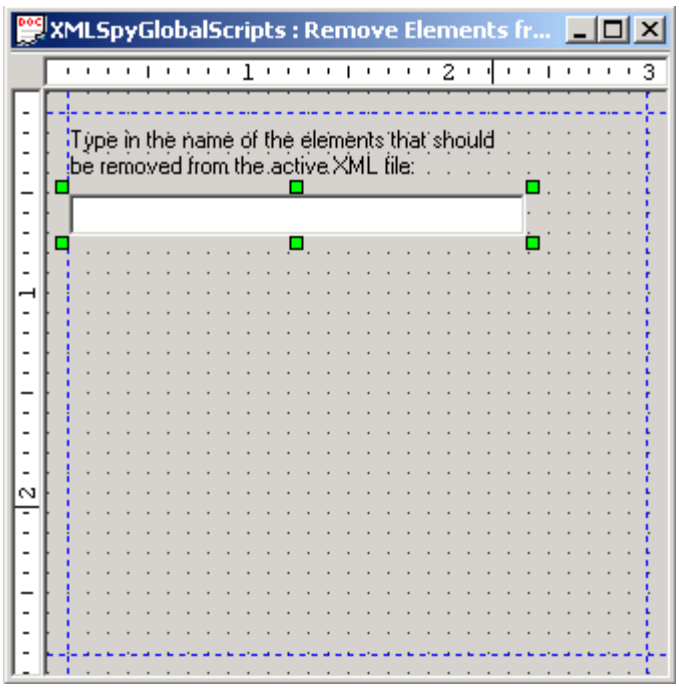
Position and dimension of a Form Object
The position and dimensions of the edit box are specified in the Property Sheet in terms of the Left, Right, Top, and Bottom properties. The Left and Right properties take values that are the distances (in pixels) of the left and right borders of the edit box, respectively, from the left border

 © 2007 Altova GmbH

of the Form. The Top and Bottom properties take values that are the distances (in pixels) of the top and bottom borders of the edit box, respectively, from the top border of the Form.

Inserting text in the Form

Add a caption for the edit box with the static text icon from the Form Objects Bar. The Design View of the Form should look something like this:



Inserting buttons

We now need two buttons in our form: **Delete** and **Cancel**. To insert these, do the following:

1. Select the button icon and then draw the two buttons in the Design View.
2. Name each button in the `( ObjectCode)` line of the respective Property Sheets: `BtnDelete` and `BtnCancel`.
3. Specify the Button text (**Delete** and **Cancel**, respectively) either by editing the button text directly in the Design View or by editing the value of the `Text` property in the Property Sheet.

Writing the required scripts

After the visual design of the Form is complete, we need to associate the Form Objects with suitable scripts. In this case, we need two types of scripts:

- A function that loops through the XML file, deleting all `XMLData` objects with a given name (specified by the user in the edit box)
- Event handlers that are executed when the two buttons are clicked

Creating the function

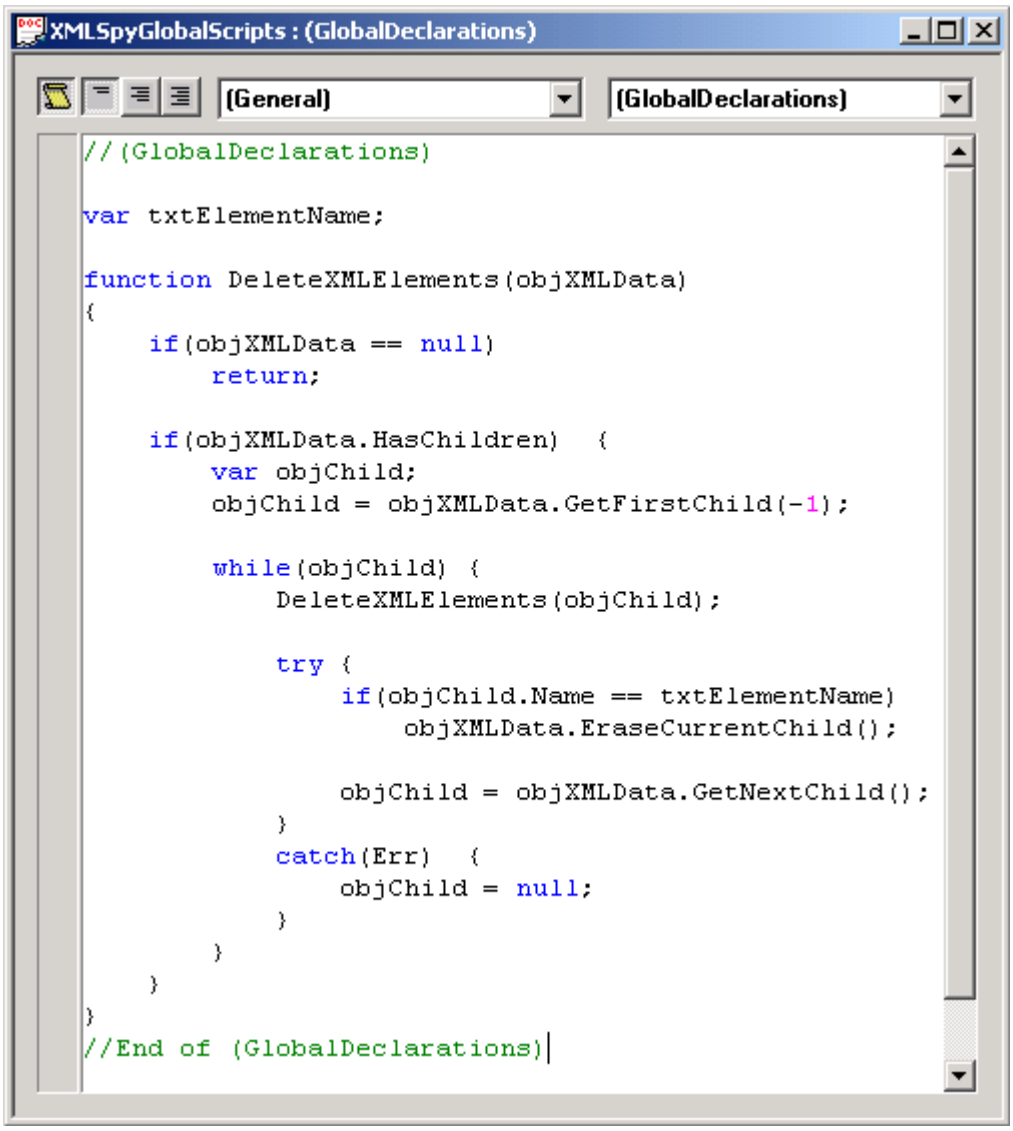
The function that deletes the `XMLData` objects must be defined in the Global Declarations module of the scripting project. Open the Global Declarations window by double-clicking the `GlobalDeclarations` module in the Project Window. Type the following code into the Global Declarations window:

```
var txtElementName;

 function DeleteXMLElements( objXMLData)
```

```
 {
   if( objXMLData == null)
     return;

   if( objXMLData. HasChildren){
     var objChild;
     objChild = objXMLData. GetFirstChild( -1);

     while( objChild) {
      DeleteXMLElements( objChild);

      try{
        if( objChild. Name == txtElementName)
        objXMLData. EraseCurrentChild( );

        objChild = objXMLData. GetNextChild( );
      }
      catch( Err) {
        objChild = null;
      }
    }
  }
}
```

The Global Declarations window should look like this:

© 2007 Altova GmbH



Creating the button events

- Select the `BtnDelete` button and click the **Events** tab in the Property Sheet.
- Select the `EventClick` property and click the ellipsis button at the right. A Script Editor window pops up with the predefined function header of the click-event. This function will be executed every time the user clicks the **Delete** button. Add the following code to the event:

```
if( EditElements. Text != "")       {
              txtElementName = EditElements. Text;
              DeleteXMLElements( Application. ActiveDocument. RootElement);
}
```

Summary of Form creation

The Form we have created using the steps above does the following:

- The **Delete** button sets the global variable `txtElementName` to the string that the user enters in the edit box (`EditElements` is the name we have assigned the edit box).
- The global variable `txtElementName` calls the function `DeleteXMLElements()` with

the root element of the active XML document. (The function `DeleteXMLElements()` is the one we have defined in the Global Declarations module.)

- The `DeleteXMLElements()` function uses the `XMLData` object of the XMLSpy API.

Running the Form

To run the Form, create a Macro as follows:

1. Open the Macros module (by double-clicking `XMLSpyMacros` in the Project Window).
2. Create a new Macro called `DeleteElements` and enter thefollowing code:

```
var a;
if( Application. ActiveDocument ! = null)
              a = Application. ShowForm( "RemoveDlg");
```

3. Run the Macro by selecting the `DeleteElements` in the Show Macros dialog, which you open by clicking **Tools | Show macros...** in the XMLSpy GUI.

Note:

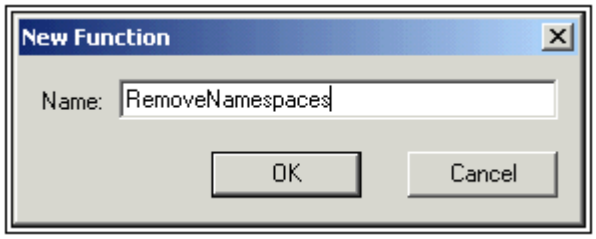
- A Form can display another Form with the function `Application. ShowForm().`

2.3.3 Writing a Macro

A Macro automates a repetitive or complex task. In this section, we write a Macro that removes all namespace prefixes from the active document.

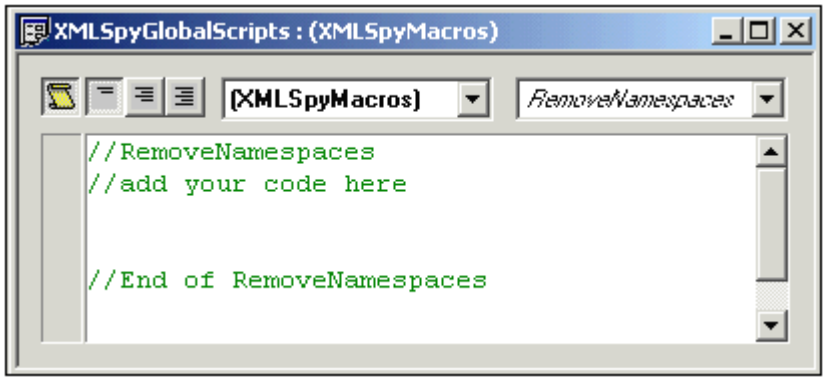
Creating a new Macro

1. To create a new Macro, right-click `XMLSpyMacros` in the Project Window and select **Add Function** from the context menu.
2. In the New Function dialog that appears, enter `RemoveNamespaces` for the name of the Macro.



The Scripting Environment creates a new Macro with this name and adds it to the `XMLSpyMacros` module.

3. Double-click the `XMLSpyMacros` module in the Project Window to open the Macros window.
4. Select the `RemoveNamespaces` Macro.



 © 2007 Altova GmbH

5. Enter the code below in the Macro window:

```
if( Application. ActiveDocument ! = null)  {

RemoveAllNamespaces( Application. ActiveDocument. RootElement);
              Application. ActiveDocument. UpdateViews();
}
```

Note: Macros do not support parameters or return values, so no function header should exist in the macro implementation.

Creating the function used in the Macro
We now have to write the `RemoveAllNamespaces` function. Creating it in the Global Declarations module makes it accessible to all Macros, Event Handlers, and Forms.

1. Activate the Global Declarations module by double-clicking it in the Project Window.
2. Append the following code for the `RemoveAllNamespaces()` function to code for any previously defined global variables or functions:

```
function RemoveAllNamespaces( objXMLData)
{
       if( objXMLData == null)
    return;

  if( objXMLData. HasChildren)      {
    var objChild;

    // spyXMLDataElement := 4
    objChild = objXMLData. GetFirstChild( 4);

    while( objChild) {
    RemoveAllNamespaces( objChild);

          try{
              var nPos, txtName;
              txtName = objChild. Name;

              if(( nPos = txtName. indexOf( ":")) >= 0)  {
                  objChild. Name = txtName. substring( nPos+1);
              }

    objChild = objXMLData. GetNextChild();
          }
          catch( Err)              {
              objChild = null;
          }
      }
    }
  }
```

This completes the creation of the new macro.

Note:

- It is possible to define local variables and helper functions within macros and event handlers. Example:

```
//return value: true allows editing
//return value: false disallows editing
var txtLocal;
function Helper()
{
  txtMessage = txtLocal;
  Application. ShowForm( "MsgBox");
```

```
}
function On_BeforeStartEditing(objXMLData)
{
  txtLocal = "On_BeforeStartEditing()";
  Helper();
}
```

- Recursive functions are supported. The function `DeleteXMLElements()` in Creating a Form, for example, calls itself recursively.

Running the Macro
Run the Macro by selecting the `DeleteElements` in the Show Macros dialog, which you open by clicking **Tools | Show macros...** in the XMLSpy GUI. See Calling macros.

Note: While a Macro is executed, Event Handlers from XMLSpy are not processed.

Modifying the Macro
This Macro can be easily modified to rename namespaces rather than remove them. You would need to carry out the following steps:

1. Design a Form (see Creating a Form) in which the user can specify the old and the new namespace names.
2. In the Macro code, change the `try-catch` block of the `RemoveAllNamespaces()` function to something like this:

```
try{
 var nPos,txtName;
 txtName = objChild.Name;

 if((nPos = txtName.indexOf(":")) >= 0){
  var txtOld;
  txtOld = txtName.substring(0,nPos);

  if(txtOld == txtOldNamespace)
   objChild.Name = txtNewNamespace + ":" + txtName.substring(nPos+1);
 }

 objChild = objXMLData.GetNextChild();
}
catch(Err){
 objChild = null;
}
```

This code assumes that `txtOldNamespace` and `txtNewNamespace` are declared as global variables and are set with the proper values.

2.3.4 Calling macros

There are two ways to call or run a Macro:

- With the **Show Macros** dialog of XMLSpy
- By creating and using a menu item (or Run command) for the Macro **in the Tools menu** of XMLSpy
- By creating and using a toolbar button for a macro

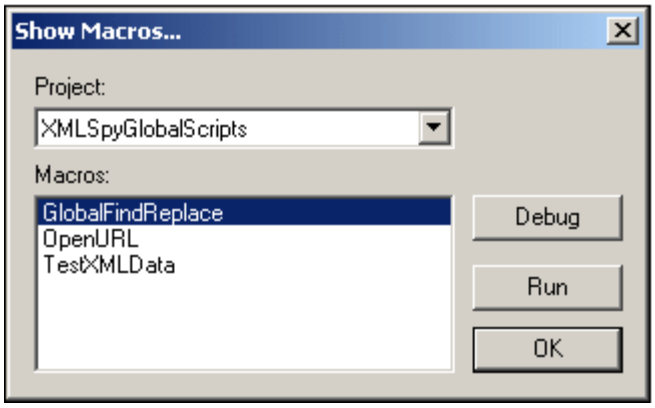
Note: While a Macro is executed, Event Handlers from XMLSpy are not processed.

Using the Show Macros dialog

1. In XMLSpy, click **Tools | Show macros**. This pops up the Show Macros dialog, which displays all macros defined in the global scripting project and in the scripting project

 © 2007 Altova GmbH

assigned to the current XMLSpy project. In the Project combo box, select whether the Macros in the global scripting project or the Macros in the assigned scripting project are to be displayed.



2. Select the Macro you wish to run.
3. Click the **Run** button. Before the Macro is executed, the Show Macros dialog is closed.

Debugging a macro
You can debug a macro from within the Scripting Environment. In the Show Macros dialog, click the **Debug** button. This pops up the Just-In-Time dialog, in which you can select from the debuggers that are installed on your system. Select a debugger and click **Yes**. The debugger starts.

Creating a menu item in the Tools menu
The XMLSpy API includes a function to add Macros as menu items to the **Tools** menu. This function can be used to add one or more Macros to the **Tools | Show macros** list of Macros. To do this carry out the following steps:

1. To add a menu item, call the XMLSpy API function `AddMacroMenuItem()`.
2. To reset the Tools menu call `ClearMacroMenu()`. This removes all previously added menu items

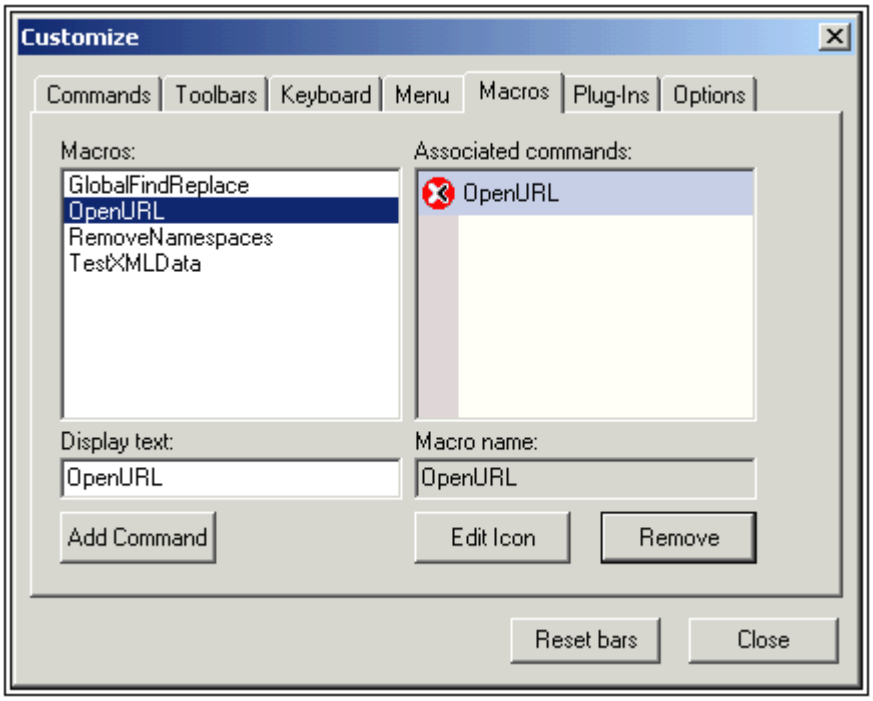
The best way to call these two functions is with the `Autorun` macro of the global scripting project or the `On_OpenProject` event.

Example:

```
Application.AddMacroMenuItem("DeleteElements","Delete Elements
   Dialog");
```

- The first parameter (`DeleteElements` in the example) is the name of the Macro. If you run the Macro and there is an open project having scripts associated with it, XMLSpy searches for the Macro in the project scripts first. If there are no project scripts, or if XMLSpy cannot find the macro, then it looks for the Macro in the global scripts.
- The second parameter (`Delete Elements Dialog`) is the display text for the menu item.

Using a toolbar icon for a macro
You can create an icon in the toolbar, which, when it is clicked, will run a Macro. To do this, click **Tools | Customize | Macros**.



Now do the following:

1. In the Macros pane, select the required Macro.
2. In the **Display text** input field enter the name of the icon. This name will appear when the cursor is placed over the icon when it is in the toolbar.
3. Click Add Command to add it to the list of commands.
4. Select the command and click Edit Icon to create a new icon.
5. Drag the finished icon from the Associated commands pane and drop it on to the toolbar when the cursor changes from an arrow to an I-beam.
6. Click Reset bars for this Macro icon and command to be displayed in the **Tools | Show Macros...** dialog.

2.3.5 Programming points to note

The following programming points should be noted:

- Out-parameters from methods of the XMLSpy API require special variables in JavaScript. Given below are some examples.

```
// use JavaScript method to access out-parameters
var strError = new Array(1);
var nErrorPos = new Array(1);
var objBadData = new Array(1);
bOK = objDoc.IsValid(strError,nErrorPos,objBadData);
```


© 2007 Altova GmbH

2.4 Menus

2.4.1 File

New Project
Creates a new project. You are prompted for the scripting language of the new project, which can be either JavaScript or VBScript. The project is opened as the active project, and its title is diaplayed in bold in the Projects Window.

Open Project
Opens an existing project. The newly opened project becomes the active project. Only one project among all the open projects can be active at a time. A project is made the active project automatically when it is opened or if it is created as a new project. Otherwise, a project is made active by explicitly specifying this—right-click a project title and select **Set as Active Project**.

If only one instance of the project is open, then its title is displayed in bold. Multiple instances of a single project can be open, and in this case the title of the currently active instance or of the last instance to have been active is displayed in bold.

Close Project
Closes the selected project. If a project is not the active project, closing it automatically makes it the active project before it is closed—therefore, after it is closed no project is active.

Save Project
Saves the selected project.

Save Project As
Saves an unnamed open project with a name or a named open project under another name.

Printing
You can print a form or script by using the Print command. A Print Preview is also available. Note that to switch back from the Print Preview to the Scripting Environment, you should press the **Close** button. Do not close the Print Preview window; this will cause the entire Scripting Environment to close.

Switch to XMLSPY...
Minimizes the Scripting Environment and restores the XMLSpy window.

Exit
Closes the Scripting Environment window.

2.4.2 Edit

Undo and Redo
The Undo and Redo commands enable you to carry out multiple undo and redo commands.

Cut, Copy, Paste, Delete
To cut, copy and delete, select an object or text fragment, and select one of these commands. To paste, place the cursor at the desired location and select the Paste command.

2.4.3 View

Toolbars
Toggles on/off a number of toolbars.

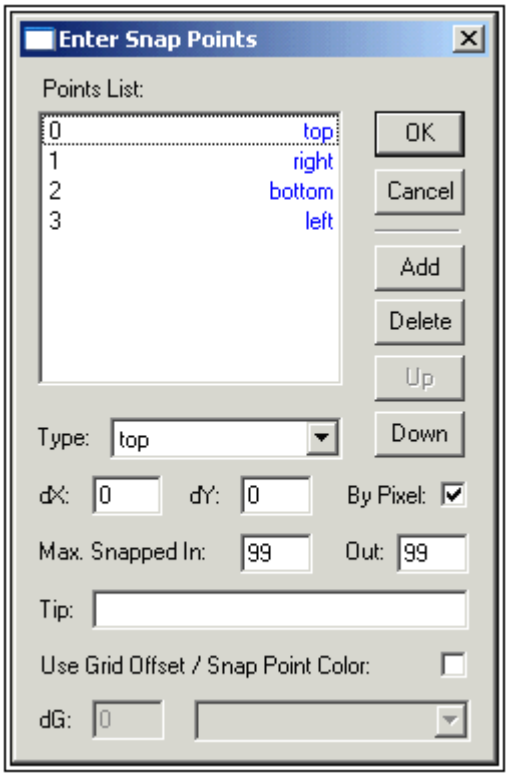
Project Bar

Toggles on/off the Project Bar.

Status Bar
Toggles on/off the Status Bar at the bottom of the Sripting Environment window.

Ruler Bars
Toggles on/off the ruler guides along the top and left margins of the page.

Snap Points
Toggles on/off the snap points of an object. Snap points are the points on an object that will snap to another object when that object is being moved. You can add and delete snap points by selecting the object, and, in the Property Sheet of the object, clicking the ellipsis in the value field of the AnchorSnaps property. This pops up the Enter Snap Ponts dialog, in which you can add snap points.



When the snap points are toggled on, they are visible and aid you to position objects accurately.

2.4.4 Project

Add Form
Enables you to add a Form to the current set of Forms in the project.

Add Function
When Events or Macros is selcted in the Project Window, you can add a Function to the list of already available functions. Selecting this command pops up the following dialog:

© 2007 Altova GmbH

Entering the name of the function and clicking OK adds a function with this name to the list of existing functions and opens a scriptig window for this function.

Remove Function
This command opens the Remove Function dialog, in which you enter the name of the Events or Macros functon to be removed.

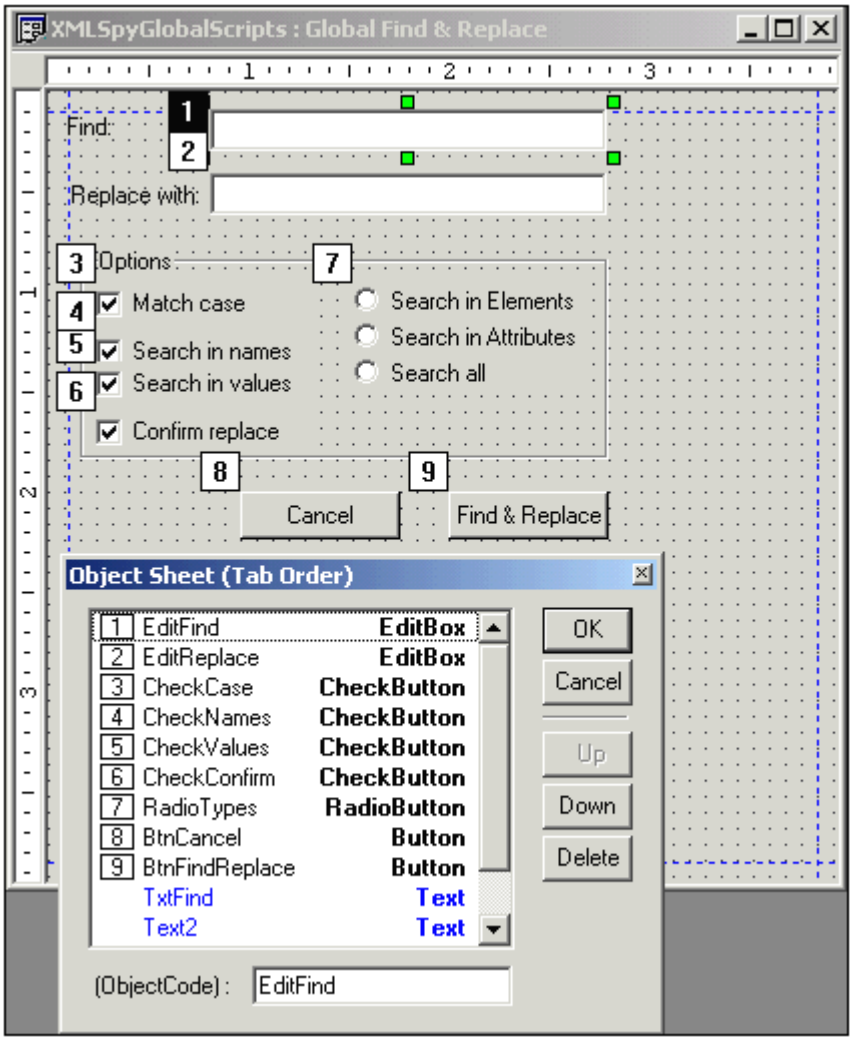
Project Information
This command gives you the path to the current project file and information about whether the project was opened from within XMLSpy or not.

2.4.5 Layout

The Layout menu contains commands for Form layout and view controls. These commands are not relevant for Modules.

Spacing, positioning, and sizing of objects
When two or more Form Objects are selected (by pressing Shift while clicking the objects), they can be spaced and sized relative to each other as well as positioned relative to the borders of the Form. You can do this by using the **Align Objects**, **Space Evenly**, **Align in View**, and **Make Same Size** commands. Note that to use the **Space Evenly** command at least three objects have to be selected.

Objects
The **Objects...** command pops up the Object Sheet (Tab Order) window, in which you specify the tab sequence of the data-input fields. Data-input fields correspond to data-input Form Objects.



When you insert a Form Object in your design, the new object is appended to the appropriate section (numbered or non-numbered) in the Object Sheet list. The Object Sheet list has two parts, a numbered part for data-input objects and a non-numbered part for display objects. The number of an object in the numbered part indicates the position of that object in the run-time tab sequence. You can change the position of an object in the tab sequence by selecting it and pressing the **Up** or **Down** buttons as required. You can also delete an object by pressing the **Delete** button.

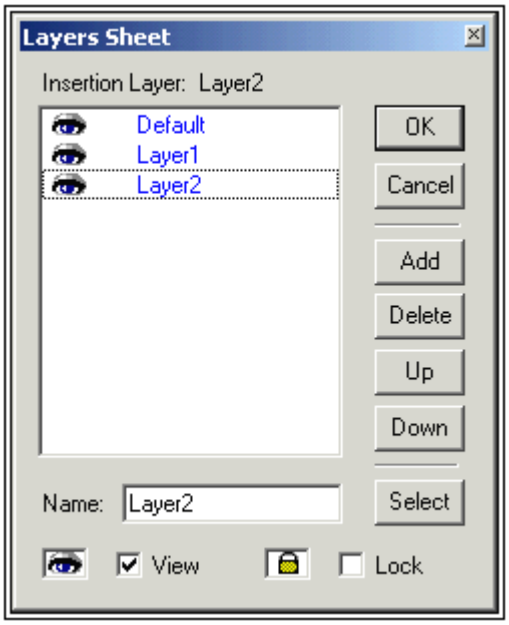
Note: The tab sequence applies to data-input objects only at run-time, i.e. when the Form is displayed for user input. In the Form Editor, pressing the Tab key first takes you through the display objects (in the order in which these are listed in the Object Sheet), and then through the data-input objects (in the tab sequence).

Layers

A Form is designed in a "Default" layer that is always the top layer. You can add additional layers below the Default layer, and move these lower layers relative to each other. The Default layer remains the top layer. Layers, therefore, allow you to add background effects to your Form.

To add and manipulate layers, select the **Layers...** command. This pops up the Layers Sheet.

© 2007 Altova GmbH

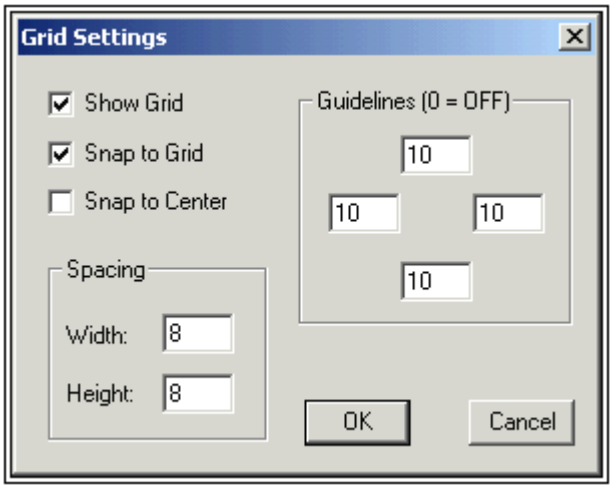


To add a new layer, click the **Add** button. You can rename a layer by selecting it and typing a new name into the **Name** input field. To delete a layer, select the layer to be deleted and click the **Delete** button. To move a layer up or down in the layer sequence, select the layer to move and press the **Up** or **Down** buttons as required. The layer sequence is ordered with the first layer in the Layers Sheet corresponding to the topmost layer of the Form.

The **Select** button selects, in the design view, all the objects of the currently selected layer. You can also turn the display of all objects in a layer on or off by using the **View** check box. Locking a layer, with the **Lock** check box, renders that layer uneditable and causes the Object Bar to be grayed out when that layer is active.

Properties
The **Properties...** command pops up the Property Sheet for the selected Form. A Property Sheet is available not only for each Form but also for each Form Object. You can modify properties of a Form or Form Object via its Property Sheet. How to use Property Sheets is described in detail in The Form Editor.

Grid Settings
The **Grid Settings...** command pops up the Grid Settings dialog, in which you can specify the appearance of the background grid for the design view.



Spacing settings specify the horizontal and vertical distances between grid lines. Four guidelines are available (top, right, bottom, left), and you can specify the offset of each from the corresponding border of the Form. Grid Settings take effect with the next design view to be made active.

Edit View
This is a toggle setting that is specified individually for each Form. Checking the **Edit View** command causes the Form to be opened in editable design view or for it to switch from non-editable view to editable design view. Unchecking the **Edit View** command causes the Form to be opened in non-editable view or for it to switch from editable design view to non-editable view. Note that the setting is made for the active Form only; it does not apply to all Forms.

Edit Script
This is a toggle setting that is specified individually for each Form. Cecking the **Edit Script** command causes a separate script window to be displayed for that Form. Unchecking the command causes the script window to close.

2.4.6 Draw

Order
This is a Form Editor command and enables you to place the selected object in front of or behind other objects on the same layer. If you wish to arrange objects which are on different layers, then you should manipulate layers in the Layers Sheet (accessed with the **Layout | Layers** command).

Zoom
This is a Form Editor command menu and enables you to zoom in and out on an object or the Form as a whole.

Group
This is a Form Editor command menu which is enabled when two or more objects, or a group of objects, are selected. It allows you to group and ungroup two or more objects. This is useful if you wish to keep objects together as a group.

Rotate
Enables you to rotate certain objects. The **Free Rotate** command is a toggle command. It allows you to grab the object by one of its handles and rotate it about its center. You acn also **Rotate Right** by 90º. The **Unrotate** command functions like an Undo command.

 © 2007 Altova GmbH

2.4.7 Window

The Window menu enables you to manipulate the arrangement of open windows in the Main Window. You can cascade and tile windows, and arrange icons. You can also use this menu to select, from a list of all open windows, the window you wish to make the active window.

3 XMLSpy Plugins

XMLSpy allows you to create your own IDE plug-ins and integrate them into XMLSpy.

Use plug-ins to:

- Configure your version of XMLSpy, add commands through menus, icons, buttons etc.
- React to events from XMLSpy.
- Run your specific code within XMLSpy with access to the complete XMLSpy API

XMLSpy expects your plug-in to implement the IXMLSpyPlugIn interface. See ATL sample files for an example using C++. See the folder "xmlspy\Examples\XMLSpyPlugIn" of your XMLSpy installation for a sample using VisualBasic.

Note: If you are interested in developing a Plugin for XMLSPY using this API, please visit the Altova Partner Portal at **http://partners.altova.com**, or contact us via e-mail at mailto:partners@altova.com to learn more about the different partnership options!

© 2007 Altova GmbH

3.1 Registration of IDE PlugIns

XMLSpy maintains a specific key in the Registry where it stores all registered IDE plug-ins:

HKEY_CURRENT_USER\Software\Altova\XML Spy\PlugIns

All values of this key are treated as references to registered plug-ins and must conform to the following format:

Value name:	ProgID of the plug-in
Value type:	must be REG_SZ
Value data:	CLSID of the component

Each time the application starts the values of the "PlugIns" key is scanned, and the registered plug-ins are loaded.

Register plug-in manually
To register a plug-in manually, use the "Customize" dialog box of the XMLSpy "Tools" menu. Use the "Add Plug-In..." button to specify the DLL that implements your plug-in. XMLSpy registers the DLL as a COM server and adds the corresponding entry in its "PlugIns" key.

If you experience problems with manual registration you can check if the CLSID of your plug-in is correctly registered in the "PlugIns" key. If this is not the case, the name of your plug-in DLL was probably not sufficiently unique. Use a different name or perform direct registration.

Register plug-in directly
A plug-in can be directly registered as an IDE plug-in by first registering the DLL and then adding the appropriate value to the "PlugIns" key of XMLSpy during plug-in setup for example. The new plug-in will be activated the next time XMLSpy is launched.

3.2 ActiveX Controls

ActiveX controls are supported. Any IDE PlugIn which is also an ActiveX control will be displayed in a Dialog Control Bar. A sample PlugIn that is also an ActiveX control is included in the `XMLSpyPlugInActiveX` folder in the `Examples` folder of your application folder.

© 2007 Altova GmbH

3.3 Configuration XML

The IDE plug-in allows you to change the user interface (UI) of XMLSpy. This is done by describing each separate modification using an XML data stream. The XML configuration is passed to XMLSpy using the GetUIModifications method of the IXMLSpyPlugIn interface.

The XML file containing the UI modifications for the IDE PlugIn, must have the following structure:

```
<ConfigurationData>
 <ImageFile>path To image file</ImageFile>
 <Modifications>
  <Modification>
   ...
  </Modification>
  ...
 </Modifications>
</ConfigurationData>
```

You can define icons or toolbar buttons for the new menu items which are added to the UI of XMLSpy by the plug-in. The path to the file containing the images is set using the ImageFile element. Each image must be 16 x 16 pixels using max. 256 colors. The image references must be arranged from left to right in a single (<ImageFile>...) line. The rightmost image index value, is zero.

The Modifications element can have any number of Modification child elements. Each Modification element defines a specific change to the standard UI of XMLSpy. Starting with version 4.3, it is also possible to remove UI elements from XMLSpy.

Structure of Modification elements
All Modification elements consist of the following two child elements:

```
<Modification>
 <Action>Type of action</Action>
 <UIElement Type="type of UI element">
 </UIElement>
</Modification>
```

Valid values for the Action element are:

Add - to add the following UI element to XMLSpy
Hide - to hide the following UI element in XMLSpy
Remove - to remove the UI element from the "Commands" list box, in the customize dialog

You can combine values of the Action element e.g. "Hide Remove"

The UIElement element describes any new, or existing UI element for XMLSpy. Possible elements are currently: new toolbars, buttons, menus or menu items. The **type** attribute, defines which UI element is described by the XML element.

Common UIElement children
The ID and Name elements are valid for all different types of XML UIElement fragments. It is however possible, to ignore one of the values for a specific type of UIElement e.g. Name is ignored for a separator.

```
<ID></ID>
<Name></Name>
```

If UIElement describes an existing element of the UI, the value of the ID element is predefined

by XMLSpy. Normally these ID values are not known to the public. If the XML fragment describes a new part of the UI, then the ID is arbitrary and the value should be less than 1000. The Name element sets the textual value. Existing UI elements can be identified just by name, for e.g. menus and menu items with associated sub menus. For new UI elements, the Name element sets the caption e.g. the title of a toolbar, or text for a menu item.

Toolbars and Menus
To define a toolbar its necessary to specify the ID and/or the name of the toolbar. An existing toolbar can be specified using only the name, or by the ID if it is known. To create a **new** toolbar both values must be set. The **type** attribute must be equal to "ToolBar".

```
<UIElement Type="ToolBar">
 <ID>1</ID>
 <Name>TestPlugIn</Name>
</UIElement>
```

To specify an XMLSpy menu you need two parameters:

- The ID of the menu bar which contains the menu. If no XML documents are open in the main window, the menu bar ID is 128. If one or more XML documents are open, the menu bar ID is 129.
- The menu name. Menus do not have an associated ID value. The following example defines the "Edit" menu of the menu bar which is active, when at least one XML document is open:

```
<UIElement Type="Menu">
 <ID>129</ID>
 <Name>Edit</Name>
</UIElement>
```

An additional element is used if you want to create a new menu. The Place element defines the position of the new menu in the menu bar:

```
<UIElement Type="Menu">
 <ID>129</ID>
 <Name>PlugIn Menu</Name>
 <Place>12</Place>
</UIElement>
```

A value of -1 for the Place element sets the new button or menu item at the end of the menu or toolbar.

Commands
If you add a new command, through a toolbar button or a menu item, the UIElement fragment can contain any of these sub elements:

```
<MacroName></MacroName>
<Info></Info>
<ImageID></ImageID>
```

If MacroName is specified, XMLSpy searches for a macro with the same name in the scripting environment and executes it each time this command is processed. The Info element contains a short description string which is displayed in the status bar, when the mouse pointer is over the associated command (button or menu item). ImageID defines the index of the icon the external image file. Please note that all icons are stored in one image file.

To define a toolbar button create an UIElement with this structure:

```
<UIElement Type="ToolBarItem">
 <!--don't reuse local IDs even the commands do the same-->
 <ID>5</ID>
```

 © 2007 Altova GmbH

```
  <Name>Open file from repository...</Name>
  <!--Set Place To -1 If this is the first button To be inserted-->
  <Place>-1</Place>
  <ImageID>0</ImageID>
  <ToolBarID>1</ToolBarID>
  <!--instead of the toolbar ID the toolbar name could be used-->
  <ToolBarName>TestPlugIn</ToolBarName>
 </UIElement>
```

Additional elements to declare a toolbar button are Place, ToolBarID and ToolBarName. ToolBarID and ToolBarName are used to identify the toolbar which contains the new or existing button. The textual value of ToolBarName is case sensitive. The (UIElement) **type** attribute must equal "ToolBarItem".

To define a menu item, the elements MenuID, Place and Parent are available in addition to the standard elements used to declare a command. MenuID can be either 128 or 129. Please see "Toolbars and Menus" for more information on these values.

The Parent element is used to identify the **menu** where the new menu entry should be inserted. As sub menu items have no unique Windows ID, we need some other way to identify the parent of the menu item.

The value of the Parent element is a path to the menu item.
The text value of the Parent element, must equal the **parent menu name** of the submenu, where the submenu name is separated by a colon. If the menu has no parent, because its not a submenu, add a colon to the beginning of the name. The **type** attribute must be set to "MenuItem". Example for an UIElement defining a menu item:

```
<UIElement Type="MenuItem">
  <!--the following element is a 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>
```

XMLSpy makes it possible to add toolbar separators and menus if the value of the ID element is set to 0.

3.4 ATL sample files

The following pages show how to create a simple XMLSpy IDE plug-in DLL using ATL. To build the DLL it is necessary to know about ATL, the wizards that generate new ATL objects, as well as MS VisualStudio.

To access the API the implementation imports the Type Library of XMLSpy. The code reads various properties and calls methods using the smart pointers provided by the #import statement.

In addition, the sample code uses the MFC class CString and the ATL conversion macros such as W2T.

At a glance the steps to create an ATL DLL are as follows:

1. Open VisualStudio and select "New..." from the "File" menu.
2. Select the "Projects" tab.
3. Select "ATL COM AppWizard" and type in a project name.
4. Select "Support for MFC" if you want to use MFC classes, or if you want to create a project for the sample code.

   Having created the project files you can add an ATL object to implement the IXMLSpyPlugIn interface:

1. Select "New ATL Object..." from the "Insert" menu.
2. Select "Simple Object" from the wizard and click "Next".
3. Type in a name for the object.
4. On the "Attributes" tab, select "Custom" for the type of interface, and disable Aggregation.

These steps produce the skeleton code for the implementation of the IDE plug-in interface. Please see the following pages on how to modify the code and achieve some basic functionality.

3.4.1 Interface description (IDL)

The IDL of the newly created ATL object contains a declaration for one COM interface.

- This interface declaration must be replaced by the declaration of IXMLSpyPlugIn as shown below.
- The IDL must also contain the definition of the SPYUpdateAction enumeration.
- Replace the generated default interface name, (created by the wizard) with "IXMLSpyPlugIn" in the coclass declaration. The IDL should then look something like the example code below:

Having created the ATL object, you then need to implement the IDE plug-in interface of XMLSpy.

```
import "oaidl.idl";
import "ocidl.idl";

// ----- please insert the following block into your IDL file -----
 typedef enum {
  spyEnable = 1,
  spyDisable = 2,
  spyCheck = 4,
  spyUncheck = 8
 } SPYUpdateAction;
```

 © 2007 Altova GmbH

```
// ----- 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 -----
```

```
 };
};
```

3.4.2 Class definition

In the class definition of the ATL object, several changes must be made. The class has to derive from IXMLSpyPlugIn, the "Interface Map" needs an entry for IXMLSpyPlugIn, and the methods of the IDE plug-in interface must be declared:

```
#ifndef __XMLSPYIDEPLUGIN_H_
#define __XMLSPYIDEPLUGIN_H_

#include "resource.h"       // main symbols

/////////////////////////////////////////////////////////////////////////////
// CXMLSpyIDEPlugIn
class ATL_NO_VTABLE CXMLSpyIDEPlugIn :
 public CComObjectRootEx<CComSingleThreadModel>,
 public CComCoClass<CXMLSpyIDEPlugIn, &CLSID_XMLSpyIDEPlugIn>,
 public IXMLSpyPlugIn
{
public:
 CXMLSpyIDEPlugIn()
 {
 }

DECLARE_REGISTRY_RESOURCEID(IDR_XMLSPYIDEPLUGIN)
DECLARE_NOT_AGGREGATABLE(CXMLSpyIDEPlugIn)

DECLARE_PROTECT_FINAL_CONSTRUCT()

BEGIN_COM_MAP(CXMLSpyIDEPlugIn)
 COM_INTERFACE_ENTRY(IXMLSpyPlugIn)
END_COM_MAP()

// IXMLSpyIDEPlugIn
public:
 virtual HRESULT _stdcall OnCommand(long nID, IDispatch* pXMLSpy);

 virtual HRESULT _stdcall OnUpdateCommand(long nID, IDispatch* pXMLSpy,
SPYUpdateAction* pAction);

 virtual HRESULT _stdcall OnEvent(long nEventID, SAFEARRAY **arrayParameters,
IDispatch* pXMLSpy, VARIANT* pReturnValue);

 virtual HRESULT _stdcall GetUIModifications(BSTR* pModificationsXML);

 virtual HRESULT _stdcall GetDescription(BSTR* pDescription);
};

#endif //__XMLSPYIDEPLUGIN_H_
```

3.4.3 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);
}
```


© 2007 Altova GmbH

3.5 IXMLSpyPlugIn

See also

Methods
OnCommand
OnUpdateCommand
OnEvent
GetUIModifications
GetDescription

Description
If a DLL is added to XMLSpy as an IDE plug-in, it is necessary that it registers a COM component that answers to an IXMLSpyPlugIn interface with the reserved uuid(88F2A622-4B7E-42CD-8D04-3C0E5389DD85), for it to be recognized as a plug-in.

3.5.1 OnCommand

See also

Declaration: `OnCommand( nID as long, pXMLSpy as IDispatch)`

Description
The `OnCommand()` method of the interface implementation, is called each time a command added by the the IDE plug-in (menu item or toolbar button) is processed. nID stores the command ID defined by the `ID` element of the respective `UIElement`.

`pXMLSpy` holds a reference to the dispatch interface of the `Application` object of XMLSpy.

Example

```
Public Sub IXMLSpyPlugIn_OnCommand(ByVal nID As Long, ByVal pXMLSpy As Object)
    If (Not (pXMLSpy Is Nothing)) Then
        Dim objDlg
        Dim objDoc As XMLSpyLib.Document
        Dim objSpy As XMLSpyLib.Application
        Set objSpy = pXMLSpy

        If nID = 3 Or nID = 5 Then
            Set objDlg = CreateObject("MSComDlg.CommonDialog")
            objDlg.Filter = "XML Files (*.xml)|*.xml|All Files (*.*)|*.*||"
            objDlg.FilterIndex = 1
            objDlg.ShowOpen

            If Len(objDlg.FileName) > 0 Then
                Set objDoc = objSpy.Documents.OpenFile(objDlg.FileName, False)
                Set objDoc = Nothing
            End If
        End If

        If nID = 4 Or nID = 6 Then
            Set objDlg = CreateObject("MSComDlg.CommonDialog")
            objDlg.Filter = "All Files (*.*)|*.*||"
            objDlg.Flags = cdlOFNPathMustExist
            objDlg.ShowSave

            If Len(objDlg.FileName) > 0 Then
                Set objDoc = objSpy.ActiveDocument

                If Not (objDoc Is Nothing) Then
                    objDoc.SetPathName objDlg.FileName
```

```
                    objDoc.Save
                    Set objDoc = Nothing
                End If
            End If
        End If

        Set objSpy = Nothing
    End If
End Sub
```

3.5.2 OnUpdateCommand

See also

Declaration: `OnUpdateCommand( nID as long, pXMLSpy as IDispatch) as SPYUpdateAction`

Description

The `OnUpdateCommand()` method is called each time the visible state of a button or menu item needs to be set. `nID` stores the command ID defined by the `ID` element of the respective `UIElement`.

`pXMLSpy` holds a reference to the dispatch interface of the `Application` object.

Possible return values to set the update state are:

```
spyEnable         = 1
spyDisable        = 2
spyCheck          = 4
spyUncheck        = 8
```

Example

```
Public Function IXMLSpyPlugIn_OnUpdateCommand( ByVal nID As Long, ByVal
pXMLSpy As Object) As SPYUpdateAction
    IXMLSpyPlugIn_OnUpdateCommand = spyDisable

    If ( Not ( pXMLSpy Is Nothing)) Then
        Dim objSpy As XMLSpyLib.Application
        Set objSpy = pXMLSpy

        If nID = 3 Or nID = 5 Then
            IXMLSpyPlugIn_OnUpdateCommand = spyEnable
        End If
        If nID = 4 Or nID = 6 Then
            If objSpy.Documents.Count > 0 Then
                IXMLSpyPlugIn_OnUpdateCommand = spyEnable
            Else
                IXMLSpyPlugIn_OnUpdateCommand = spyDisable
            End If
        End If
    End If
End Function
```

3.5.3 OnEvent

See also

Declaration: `OnEvent( nEventID as long, arrayParameters as SAFEARRAY( VARIANT), pXMLSpy as IDispatch) as VARIANT`

© 2007 Altova GmbH

Description

`OnEvent()` is called each time an event is raised from XMLSpy.

Possible values for `nEventID` are:

On_BeforeStartEditing	= 1
On_EditingFinished	= 2
On_FocusChanged	= 3
On_Beforedrag	= 4
On_BeforeDrop	= 5
On_OpenProject	= 6
On_OpenDocument	= 7
On_CloseDocument	= 8
On_SaveDocument	= 9

Events available since XMLSpy 4r4:

On_DocEditDragOver	= 10
On_DocEditDrop	= 11
On_DocEditKeyDown	= 12
On_DocEditKeyUp	= 13
On_DocEditKeyPressed	= 14
On_DocEditMouseMove	= 15
On_DocEditButtonUp	= 16
On_DocEditButtonDown	= 17
On_DocEditContextMenu	= 18
On_DocEditPaste	= 19
On_DocEditCut	= 20
On_DocEditCopy	= 21
On_DocEditClear	= 22
On_DocEditSelectionChanged	= 23

Events available since XMLSpy 2004:

On_DocEditDragOver	= 10

Events available since XMLSpy 2004r4 (type library version 1.4):

On_BeforeOpen Project	= 25
On_BeforeOpenDocument	= 26
On_BeforeSaveDocument	= 27
On_BeforeCloseDocument	= 28
On_ViewActivation	= 29
On_DocEditKeyboardEvent	= 30
On_DocEditMouseEvent	= 31

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.

3.5.4 GetUIModifications

See also

Declaration: `GetUIModifications()` as String

Description

The `GetUIModifications()` method is called during initialization of the plug-in, to get the configuration XML data that defines the changes to the UI of XMLSpy. The method is called when the plug-in is loaded for the first time, and at every start of XMLSpy.

See also Configuration XML for a detailed description how to change the UI.

Example

```
Public Function IXMLSpyPlugIn_GetUIModifications() As String
    ' GetUIModifications() gets the XML file with the specified modifications
of
    ' the UI from the config.xml file in the plug-in folder
    Dim strPath As String
    strPath = App.Path

    If Len(strPath) > 0 Then
        Dim fso As New FileSystemObject
        Dim file As file

        Set file = fso.GetFile(strPath & "\config.xml")

        If (Not (file Is Nothing)) Then
            Dim stream As TextStream
```

 © 2007 Altova GmbH

```
            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
```

3.5.5 GetDescription

See also

Declaration: `GetDescription()` as String

Description

`GetDescription()` is used to define the description string for the plug-in entries visible in the Customize dialog box.

Example

```
Public Function IXMLSpyPlugIn_GetDescription() As String
    IXMLSpyPlugIn_GetDescription = "Sample Plug-in for XMLSpy;This Plug-in
demonstrates the implementation of a simple VisualBasic DLL as a Plug-in for
XMLSpy."
End Function
```

4 The XMLSpy API

The COM-based XMLSpy API of XMLSpy enables other applications to use the functionality of XMLSpy. As a result, it is now possible to automate a wide range of tasks, from validating an XML file to modifying complex XML content (with the XMLData interface). The XMLSpy API is also used by the built-in Scripting Environment of XMLSpy to access application functionality (for a complete description of which, see Scripting). It is also used by IDE Plugins.

XMLSpy and the XMLSpy API follow the common specifications for automation servers set out by Microsoft. It is possible to access the methods and properties of the XMLSpy API from common development environments, such as those using C, C++, VisualBasic, and Delphi, and with scripting languages like JavaScript and VBScript.

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.
- Terminate XMLSpy with the Application.Quit method.
- Don't forget to disable dialogs if the user interface is not visible.
- See Error handling for details of how to avoid annoying error messages.
- Free references explicitly if you are using C or C++.

This documentation
The documentation about the XMLSpy API is broadly divided into two parts.

- The first part consists of an Overview, which describes the object model for the API, important concepts, and Usage Examples to help you get started.
- The second part is a reference section that contains descriptions of all the interface objects of the XMLSpy API.

There have been improvements and updates made from release to release. The differences between releases are documented in Release Notes.

© 2007 Altova GmbH

4.1 Overview

This overview of the XMLSpy API provides you with the object model for the API, and with a description of the most important API processes. The following topics are covered:

- The object model
- Simple document access
- Error handling
- Events
- Import and export of data
- Using XMLData to modify document structure
- The DOM and XMLData

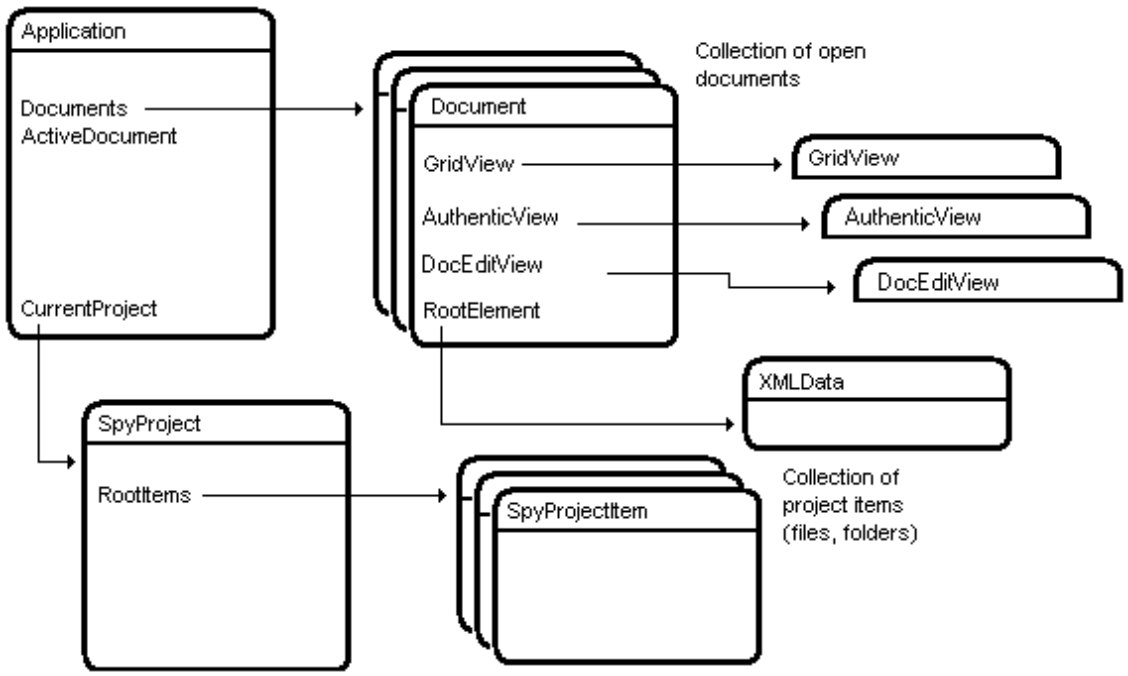
Changes to the API from from version to version of the type library are listed in the release notes section.

4.1.1 Object model

The starting point for every application which uses the XMLSpy API is the `Application` object. This object contains general methods like import/export support and references to the open documents and any open project.

To create an instance of the `Application` object, call `CreateObject("XMLSpy.Application")` from VisualBasic or a similar function from your preferred development environment to create a COM object. There is no need to create any other objects, to use the complete XMLSpy API (It is in fact not even possible). All other interfaces are accessed through other objects with the Application object as the starting point.

The picture below shows you the links between the main objects of the XMLSpy 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 of 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. Most of the times you either open a project and access the documents from there or you directly open a document via the Documents interface.

4.1.2 Simple document access

Create and open files
Use the methods of the Documents interface to create and open documents. This interface is accessible via the Application.Documents property.

Example:

```
Dim objDoc As Document
Set objDoc = objSpy.Documents.OpenFile("C:\xmlfiles\myfile.xml", False)
```

Sometimes it is necessary to show a file dialog to the user to select a file. This usually happens in the scripting environment and you only need to pass True as the second parameter to OpenFile().

To create a new file use Documents.NewFile(). Any existing file with the same path will be overwritten during the next Document.Save() call.

Documents collection
All open documents are accessible via the Documents collection. All collection objects in the XMLSpy API conform to the Microsoft specifications. So it is possible to use the For-Each loop in VisualBasic to iterate through the open documents.

Example:

```
Dim objDocs As Documents
Dim objDoc As Document

Set objDocs = objSpy.Documents

For Each objDoc In objDocs
 'do something useful with your document
 objDoc.AssignXSL "C:\myXSL.xsl", False
Next
```

Another way to access a document is the Documents.Item method. The method takes an index as a parameter and gets the corresponding document object from the collection. Please note that 1 is the first index value. Documents.Count retrieves the total number of documents.

Example:

```
Dim objDocs As Documents
```

 © 2007 Altova GmbH

```
Dim objDoc As Document

Set objDoc = objDocs.Item(1) 'gets the first document
```

Validation
One common task on documents is to validate them against an assigned schema or DTD. If the XML file has no schema or DTD already assigned, use Document.AssignSchema or Document.AssignDTD to add the necessary references to the document.

Examples:

```
objSpy.ActiveDocument.AssignSchema "C:\mySchema.xsd", False

 Or

objSpy.ActiveDocument.AssignDTD "C:\myDTD.dtd", False
```

If you want the user to select a schema or DTD, pass True as the second parameter to these functions to display a file-dialog. These methods only put the reference into the document and do not check the existence of the specified file. If the file path is not valid, the validation will fail.

After you have assigned a valid schema or DTD reference to your file, you are able to validate it with Document.IsValid. IsValid needs some out-parameters that must be declared as VARIANTs to be accessible from script languages like VBScript and JavaScript.

Example:

```
Dim bValid   As Boolean
Dim strMsg   As Variant
Dim nPos  As Variant
Dim objBadXMLData As Variant

bValid = objSpy.ActiveDocument.IsValid(strMsg, nPos, objBadXMLData)

If bValid = False Then
 a = MsgBox("The document is not valid:" & Chr(13) &
   strMsg & Chr(13) & "position: " & nPos &
   Chr(13) & "XMLData name: " & objBadXMLData.Name)
Else
 a = MsgBox("The document is valid")
End If
```

4.1.3 Error handling

The XMLSpy API returns errors in two different ways. Every API method returns an HRESULT. This return value informs the caller about any malfunctions during the execution of the method. If the call was successful, the return value is equal to S_OK. C/C++ programmers generally use HRESULT to detect any errors.

VisualBasic, scripting languages and other high-level development environments, don't give the programmer access to the returning HRESULT of a COM call. They use the second error raising mechanism also supported by the XMLSpy API, the IErrorInfo interface. If an error occurs the API creates a new object that implements the IErrorInfo interface. The development environment takes this interface, and fills its own error handling mechanism with the provided information.

The next paragraphs describes how to deal with errors raised from the XMLSpy API in different development environments.

VisualBasic
A common way to handle errors in VisualBasic is to define an error handler. This error handler can be set with the `On Error` statement. Usually the handler displays a error message and does some cleanup to avoid spare references and any kind of resource leaks. VisualBasic fills its own `Err` object with the information from the `IErrorInfo` interface.

Example:

```
Sub Validate()
 'place variable declarations here

 'set error handler
 On Error GoTo ErrorHandler

 'if IsValid() fails, program execution continues
            'at ErrorHandler:
 bValid = objSpy.ActiveDocument.IsValid(strMsg,nPos,objBadXMLData)

 'additional code comes here

 'exit
 Exit Sub

 ErrorHandler:
 Set objBadXMLData = Nothing

 MsgBox("Error: " & (Err.Number - vbObjectError) & Chr(13) &
  "Description: " & Err.Description)
End Sub
```

JavaScript
The Microsoft implementation of JavaScript (JScript) provides a try-catch mechanism to deal with errors raised from COM calls. It is very similar to the VisualBasic approach, because you also declare an error object containing the necessary information.

Example:

```
Function Validate()
{
 // please insert variable declarations here

 try{
  bValid = objSpy.ActiveDocument.IsValid(sMsg,nPos,objBad);
 }
 catch(Error) {
  objBad = Null;
  sError = Error.description;
  nErrorCode = Error.number;
  Return False;
 }

 Return bValid;
}
```

C/C++
C/C++ gives you easy access to the `HRESULT` of the COM call and to the `IErrorInterface`.

```
HRESULThr

// Call IsValid() from the XMLSpy API
If(FAILED(hr = ipDoc->IsValid(&varMsg,&varPos,&varBadData)))      {
 IErrorInfo *ipErrorInfo = Null;
```

 © 2007 Altova GmbH

```
  If( SUCCEEDED(::GetErrorInfo(0,&ipErrorInfo)))       {
   BSTRbstrDescr;
   ipErrorInfo->GetDescription(&bstrDescr);

   // handle Error information
   wprintf(L"Error message:\t%s\n",bstrDescr);
   ::SysFreeString(bstrDescr);

   // release Error info
   ipErrorInfo->Release();
  }
 }
```

4.1.4 Events

The automation interface of XMLSPY provides a large set of events to allow for a tight integration of XMLSPY and its clients. The way events can be used by clients depends on their technology used to connect to XMLSPY and on the clients programming language.

XMLSPY scripting environment - macros, forms and event handlers written with the FormEditor

The XMLSPY scripting environment automatically invokes the event handler with the corresponding name. Use the FormEditor to add code for those events you want to handle. The FormEditor predefines the correct signatures for all event handlers when you create a new project. For more information see the scripting environment.

VBScript

All event handler functions start with 'On_'. To stay compatible to existing events, their name might differ slightly from the name used on the connection point interface. Specification of event parameters is optional so that old event handlers that did not expect any parameters still work.

JScript

All event handler functions start with 'On_'. To stay compatible to existing events, their name might differ slightly from the name used on the connection point interface. Specification of event parameters is optional so that old event handlers that did not expect any parameters still work.

XMLSPY IDE Plugin

XMLSPY Plugins get informed about outstanding events by a call to the method IXMLSpyPlugIn.OnEvent. Different events are identified by a unique number. Parameters are passed to the event handler packed into a save-array.
Alternatively, Plugins can use the connection point mechanism to request events on specific objects like any other external client. See below for more information.

External clients

COM specifies the connection point mechanism to define a way how a client can register itself at a server for callbacks. 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.

VBA - VisualBasic for Applications

Use the Dim WithEvents statement to receive events from a dedicated object instance.

```
' tell VBA to connect to events once this objects gets created
Dim WithEvents objView As XMLSpyLib.AuthenticView
' initialize the object somewhere in your code
...
Set objView = objSpy.ActiveDocument.AuthenticView
...
```

```
' sample event handler for the OnMouseEvent of the object in objView
Private Function objView_OnMouseEvent ( ByVal i_nXPos As Long, ByVal
i_nYPos As Long,
                                        ByVal i_eMouseEvent As
XMLSpyLib.SPYMouseEvent,
                                        ByVal i_pRange As
XMLSpyLib.IAuthenticRange ) As Boolean
    If (i_eMouseEvent = (XMLSpyLib.spyLeftButtonDownMask Or
XMLSpyLib.spyCtrlKeyDownMask)) Then
        On Error Resume Next
        i_pRange.Select
        objView_OnMouseEvent = True
    Else
        objView_OnMouseEvent = False
    End If
End Function
```

Windows Scripting Technologies - VBScript

Use the method WScript.ConnectObject to receive events.

```
' 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")

' create document object and connect to its events
objSpy.Visible = True
Set objDoc = objSpy.Documents.OpenFile ("C:\\Program
Files\\Altova\\XMLSPY2004\\Examples\\OrgChart.xml", 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
```

Windows Scripting Technologies - JScript

Use the method WScript.ConnectObject to receive events.

```
// the event handler function
function DocEvent_OnBeforeCloseDocument(objDocument)
{
   WScript.Echo("received event - before closing document");
}

// create or connect to XMLSPY
try
{
   // create the environment and XMLSPY
   objWshShell = WScript.CreateObject("WScript.Shell");
   objFSO = WScript.CreateObject("Scripting.FileSystemObject");
   objSpy = WScript.GetObject("", "XMLSpy.Application");
}
catch(err)
   { WScript.Echo ("Can't create WScript.Shell object or XMLSPY"); }
```

 © 2007 Altova GmbH

```
// create document object and connect to its events
objSpy.Visible = true;
objDoc = objSpy.Documents.OpenFile ("C:\\Program
Files\\Altova\\XMLSPY2004\\Examples\\OrgChart.xml", false);
WScript.ConnectObject(objDoc, "DocEvent_");

// keep running while waiting on the event
// in the meantime close this document in XMLSPY manually
WScript.Echo ("sleeping for 10 seconds ...");
WScript.Sleep (10000);

objDoc = null;
WScript.Echo ("stopped listening for event");
objSpy.Quit();
```

C++

The following is an excerpt of a C++ client using connection points via ATL

```
#import "XMLSpy.tlb"
using namespace XMLSpyLib;

static const int XMLSPYSE_APPLICATION_SOURCE_ID = 0;
class XMLSpySEApplicationEventPump;

// nested class to receive application events
typedef IDispEventImpl< XMLSPYSE_APPLICATION_SOURCE_ID,
                        XMLSpySEApplicationEventPump,
                        &DIID__IApplicationEvents,
                        &LIBID_XMLSpyLib, TYPELIB_MAJ, TYPELIB_MIN >
          XMLSpySEApplicationEventImpl;

// the class receiving the application events using a sink map
class XMLSpySEApplicationEventPump :
   public XMLSpySEApplicationEventImpl
{
   private:
          static _ATL_FUNC_INFO Info_OnBeforeOpenProject;

   public:
          XMLSpySEApplicationEventPump ();
          ~XMLSpySEApplicationEventPump();

   // connection point events on XMLSpySEApplicationEventImpl
   public:
          void __stdcall OnBeforeOpenProject (IFileSelectionDlg
*io_ipDlg);

   BEGIN_SINK_MAP(XMLSpySEApplicationEventPump)
          // OnBeforeOpenProject
          SINK_ENTRY_INFO(XMLSPYSE_APPLICATION_SOURCE_ID,
DIID__IApplicationEvents,
                                  1, OnBeforeOpenProject,
&Info_OnBeforeOpenProject)
   END_SINK_MAP()
};

// we use here explicit declaration of event signature to avoid loading
the typelib
_ATL_FUNC_INFO XMLSpySEApplicationEventPump::Info_OnBeforeOpenProject =
   {CC_STDCALL, VT_EMPTY, 1, {VT_DISPATCH}};

XMLSpySEApplicationEventPump::XMLSpySEApplicationEventPump ()
{
   ...
```

```
   // get an IUnknown pointer to XMLSpy application from somewhere and
store it in ipXMLSPY
   HRESULT hRes = XMLSpySEApplicationEventImpl::DispEventAdvise( ipXMLSPY,
&DIID__IApplicationEvents);
   ...
}

XMLSpySEApplicationEventPump::~XMLSpySEApplicationEventPump()
{
   ...
   // disconnect from connection point
   HRESULT hRes =
XMLSpySEApplicationEventImpl::DispEventUnadvise( ipXMLSPY,
&DIID__IApplicationEvents);
   ...
}

// the handler for the OnBeforeOpenProject event
void __stdcall XMLSpySEApplicationEventPump::OnBeforeOpenProject
(IFileSelectionDlg *io_ipDlg)
{
   // do something with io_ipDlg here
}
```

4.1.5 Import and export of data

Before you implement your import and export tasks with the XMLSpy API, it is a good practice to test the connections, parameters, SQL queries and so on in XMLSpy. This way, you are able to verify the results and to make quick adjustments to all import or export parameters.

Most of the methods for importing and exporting data are placed in the `Application` object, the remaining functions are accessible via the `Document` interface.

There is some preparatory work necessary, before the actual import or export can be started. Every import/export job consists of two parts. You need to define a connection to your data and the specific behaviour for the import/export process.

In case of an import, the connection is either a database, a text-file or a Word document. The behaviour is basically which data (columns) should be imported in XMLSpy.

In case of an export, the connection is either a database or a text file. Specify which data (elements of the XML file) and additional parameters (e.g. automatic key generation or number of sub-levels) to use from the XML-structure for the behaviour.

The properties in the `DatabaseConnection`, `TextImportExportSettings` and `ExportSettings` interfaces have default values. See the corresponding descriptions in the `Interfaces` chapter for further information.

Import from database
These are the steps to establish a connection to an existing database for import:

1. Use a `DatabaseConnection` object and set the properties:
   The method `Application.GetDatabaseSettings` returns a new object for a database connection:

```
Dim objImpSettings As DatabaseConnection
Set objImpSettings = objSpy.GetDatabaseSettings
```

You have to set either an **ADO connection string**,

- `objImpSettings.ADOConnection = strADOConnection`

 © 2007 Altova GmbH

or the **path** to an existing database file:

- `objImpSettings.File = "C:\myDatabase.mdb"`

To complete the settings you create a **SQL select statement** to define the data to be queried:

- `objImpSettings.SQLSelect = "SELECT * FROM myTable"`

2. Call `Application.GetDatabaseImportElementList` to get a collection of the resulting columns of the SQL query:

```
Dim objElementList As ElementList
Set objElementList =
objSpy.GetDatabaseImportElementList(objImpSettings)
```

This collection gives you the opportunity to control which columns should be imported and what type the new elements will become. Each item of the collection represents one column to import. If you remove an item, the corresponding column will not be imported. You can additionally modify the `ElementListItem.ElementKind` property, to set the type of the created 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 (Nothing in VisualBasic) as the second parameter to `ImportFromDatabase()` to import all columns as plain XML elements.

3. Start the import with `Application.ImportFromDatabase`:

```
Dim objImpDoc As Document
Set objImpDoc =
objSpy.ImportFromDatabase(objImpSettings,objElementList)
```

Import from Text

Importing data from a text file is similar to the import from a database. You must use other interfaces (described in steps 1-3 below) with different methods and properties:

1. Use a `TextImportExportSettings` object and set the properties:
The method `Application.GetTextImportExportSettings` returns a new object to specify a text file for import.

```
Dim objImpSettings As TextImportExportSettings
Set 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 = "C:\myFile.txt"
objImpSettings.HeaderRow = False
```

Call `Application.GetTextImportElementList` to get a collection of all columns inside the text file:

```
Dim objElementList As ElementList
Set objElementList = objSpy.GetTextImportElementList(objImpSettings)
```

3. Start the import with `Application.ImportFromText`:

```
Dim objImpDoc As Document
Set objImpDoc = objSpy.ImportFromText(objImpSettings,objElementList)
```

Export to database

1. Use a DatabaseConnection object and set the necessary properties.
   All properties except SQLSelect are important for the export. ADOConnection or File defines the target for the output. You need to set only one of them.

2. Fill an ExportSettings object with the required values.
   These properties are the same options as those available in the export dialog of XMLSpy. Select the menu option **Convert | Export to Text files/Database** to see the options and try a combination of export settings. After that it is easy to transfer these settings to the properties of the interface.

   Call "Application.GetExportSettings" to get a ExportSettings object:

   ```
   Dim objExpSettings As ExportSettings
   Set objExpSettings = objSpy.GetExportSettings

   objExpSettings.CreateKeys = False
   objExpSettings.ExportAllElements = False
   objExpSettings.SubLevelLimit = 2
   ```

3. Build an element list with Document.GetExportElementList.
   The element list enables you to eliminate XML elements from the export process. It also gives you information about the record and field count in the RecordCount and FieldCount properties. Set the ExportSettings.ElementList property to this collection. It is possible to set the element list to null/Nothing (default) to export all elements.

4. Call Document.ExportToDatabase to execute the export.
   The description of the ExportToDatabase method contains also a code example for a database export.

Export to text

1. Use a TextImportExportSettings object and set the necessary properties.

2. Fill an ExportSettings object with the required values.
   See item number 2 from "Export to database" earlier on this page.

3. Build an element list with Document.GetExportElementList.
   See item number 3 from "Export to database" earlier on this page.

4. Call Document.ExportToText to execute the export.
   The description of the ExportToText method contains also a code example for a database export.

4.1.6 Using XMLData to modify document structure

XMLData gives you access to the elements of an currently open XML file. It enables you to perform all necessary modifications to the elements of the XML structure. The main functionality of XMLData is:

1. Access to the names and values of all kinds of elements (e.g. elements, attributes)

 © 2007 Altova GmbH

2. Creation of new elements of all kinds.
3. Insertion and appending of new elements.
4. Erasing of existing child elements.

Structure of XMLData
Before you can use the `XMLData` interface, you have to know how an existing XML file is mapped into a `XMLData` structure. One major thing you must be aware of is, that `XMLData` has no separate branch of objects for attributes.

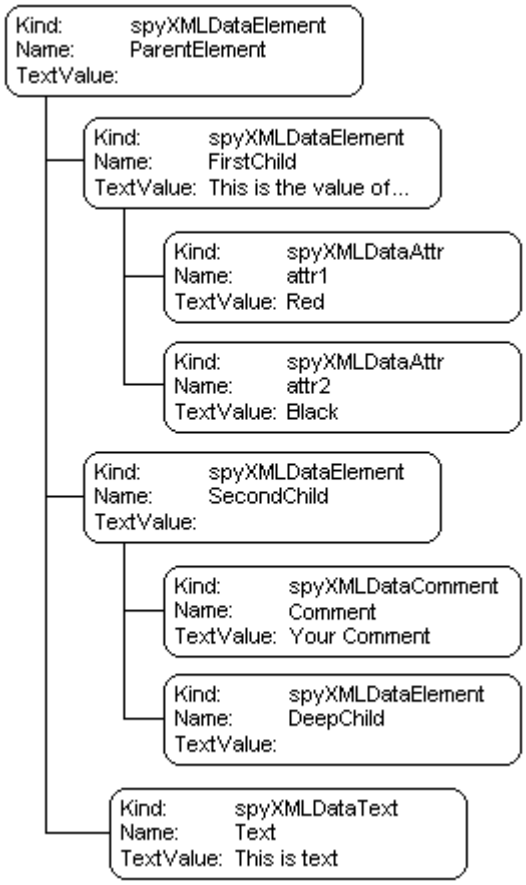
The attributes of an element are also children of the element. The `XMLData.Kind` property, gives you the opportunity to distinguish between the different types of children of an element.

Example:

This XML code,

```
<ParentElement>
 <FirstChild attr1="Red" attr2="Black">
  This is the value of FirstChild
 </FirstChild>
 <SecondChild>
  <!--Your Comment-->
  </DeepChild>
 </SecondChild>
 This is Text
</ParentElement>
```

is mapped to the following XMLData object structure:



The parent of all XML elements inside of a document is the property `Document.RootElement`. The first element of a document's content - without the XML prolog - is accessible with the property `DocumentDataRoot`. Use one of these `XMLData` objects to get references to all other XML elements in the structure.

Name and value of elements
To get and to modify the name and value of all types of XML elements use the `XMLData.Name` and `XMLData.TextValue` properties. It is possible that several kinds of `XMLData` objects and empty elements do not have a text value associated.

Creation and insertion of new XMLData objects
The creation of a new XML language entity requires the following steps:

1. Create the new `XMLData` object:
   Use the `Document.CreateChild` method to create a new `XMLData` object. Set name and value after you have inserted the new XML entity (see point 3).
2. Find the correct location for the new XMLData object:
   To insert a new `XMLData` object you have to get a reference to the parent first. If the new child is to become the last child of the parent, use the `XMLData.AppendChild` method to insert the `XMLData` object. If the new child should be located elsewhere in the sequence of child objects, use the `XMLData.GetFirstChild` and `XMLData.GetNextChild` to move the iterator to the child before which the new child should be inserted.
3. Insert the new child with `XMLData.InsertChild`. The new child will be inserted immediately before the current child.

 © 2007 Altova GmbH

The following example adds a third child between <FirstChild> and the <SecondChild> element:

```
Dim objParent As XMLData
Dim objChild As XMLData
Dim objNewChild As XMLData

Set objNewChild = objSpy.ActiveDocument.CreateChild(spyXMLDataElement)

'objParent is set to <ParentElement>
'GetFirstChild(-1) gets all children of the parent element
'and move to <SecondChild>
Set objChild = objParent.GetFirstChild(-1)
Set objChild = objParent.GetNextChild

objParent.InsertChild objNewChild
objNewChild.Name = "OneAndAHalf"
Set objNewChild = Nothing
```

Child elements should be inserted in a special order. Please avoid inserting attributes into a sequence after any other child elements, i.e. make sure that attributes are placed first.

Copying of existing XMLData objects

If you want to insert existing XMLData objects at a different place in the same document or into another document you can't use the XMLData.InsertChild and XMLData.AppendChild methods. These methods only work for new XMLData objects.

Instead of using InsertChild or AppendChild you have to copy the object hierarchy manually. The following function written in JavaScript is an example for recursively copying XMLData:

```
// ------------------------------------------------------------
// XMLSpy scripting environment - GlobalDeclarations - JScript
// return a deep copy of the XMLData Object.
// the new object is owned by the document sepcified in objDoc
// ------------------------------------------------------------
function XMLDataDeepCopy(objXMLData, objDoc)
{
 // create new element of same kind
 var objNew = objDoc.CreateChild(objXMLData.Kind);

 // copy name, some element have default names and do not
 // allow to modifying these names.
 try
 { objNew.Name = objXMLData.Name; }
 catch (e)
 {
  if ((e.number & 0xffff) != 1513)      // modification operation not allowed
   throw(e);
 }

 // copy the text value
 objNew.TextValue = objXMLData.TextValue;

 // copy and insert all children
 if(objXMLData.HasChildren)
 {
  var objChild = objXMLData.GetFirstChild(-1);

  while(objChild)
  {
   try
```

```
    {
     objNew.AppendChild(XMLDataDeepCopy(objChild, objDoc));
     objChild = objXMLData.GetNextChild();
    }
    catch(e)
    {
     if ((e.number & 0xffff) == 1503)     // end of list
      objChild = null;
     else
      throw(e);
    }
   }
  }

  return objNew;
}
```

Erasing of XMLData objects

`XMLData` provides two methods for the deletion of child objects, `XMLData.EraseAllChildren` and `XMLData.EraseCurrentChild`.

To erase `XMLData` objects you need access to the parent of the elements you want to remove. Use `XMLData.GetFirstChild` and `XMLData.GetNextChild` to get a reference to the parent `XMLData` object.

See the method descriptions of `EraseAllChildren` and `EraseCurrentChild` for examples how to erase XML elements.

4.1.7 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) {
```


© 2007 Altova GmbH

```
  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;
}
```

4.1.8 Obsolete Authentic 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 as a table. When represented as a table, rows

 © 2007 Altova GmbH

and their contents can be manipulated individually, thereby allowing you to manipulate each of the repeatable elements individually. Such row operations would be performed by an external script.

If an external script is to perform row operations then two steps must occur:

- The first step checks whether the cursor is currently in a row using a property. Such a check could be, for example, `IsRowInsertEnabled`, which returns a value of either TRUE or FALSE.
- If the return value is TRUE then a row method, such as `RowAppend`, can be called. (`RowAppend` has no parameters and returns no value.)

The following is a list of properties and methods available for table operations. Each property returns a BOOL, and the methods have no parameter.

Property	Method	Table operations
`IsRowInsertEnabled`	**RowInsert**, superseded by `AuthenticRange.InsertRow`	Insert row operation
`IsRowAppendEnabled`	**RowAppend**, superseded by `AuthenticRange.AppendRow`	Append row operation
`IsRowDeleteEnabled`	**RowDelete**, superseded by `AuthenticRange.DeleteRow`	Delete row operation
`IsRowMoveUpEnabled`	**RowMoveUp**, superseded by `AuthenticRange.MoveRowUp`	Move XML data up one row
`IsRowMoveDownEnabled`	**RowMoveDown**, superseded by `AuthenticRange.MoveRowDown`	Move XML data down one row
`IsRowDuplicateEnabled`	**RowDuplicate**, superseded by `AuthenticRange.DuplicateRow`	Duplicate currently selected row

4.1.9 Obsolete Authentic Editing operations

When XML data is displayed as data in Authentic, it is possible to manipulate individual elements using standard editing operations such as cut, copy, and paste. However, not all XML data nodes can be edited. So, in order to carry out an editing operation, first a property is used to test whether editing is possible, and then a method is called to perform the editing operation.

The only method that does not have a test is the method `EditSelectAll`, which automatically selects all elements displayed in the document.

The following is a list of properties and methods that perform editing operations. Each property returns a BOOL, and the methods have no parameter.

Property	Method	Editing operation
`IsEditUndoEnabled`	**EditUndo**, superseded by `AuthenticView.Undo`	Undo an editing operation
`IsEditRedoEnabled`	**EditRedo**, superseded by `AuthenticView.Redo`	Redo an editing operation
`IsEditCopyEnabled`	**EditCopy**, superseded by `AuthenticRange.Copy`	Copy selected text to the clipboard

IsEditCutEnabled	**EditCut**, superseded by AuthenticRange.Cut	Cut selected text to the clipboard
IsEditPasteEnabled	**EditPaste**, superseded by AuthenticRange.Paste	Paste from clipboard to current cursor position
IsEditClearEnabled	**EditClear**, superseded by AuthenticRange.Delete	Clear selected text from XML document

© 2007 Altova GmbH

4.2 Usage Examples

The automation interface of XMLSpy can be used in many different ways and accessed from different programming languages. In this section you will find sample code that show the capabilities of the new AuthenticView and AuthenticRange interfaces. These samples should make it easier for you to get started with your own project. Although they are more complex then those you will find in the descriptions of the various methods and properties, they are fragments and might require additional code or small adaptations in order for them to be properly executed in your environment.

The listed code samples are:

- JScript: Bubble Sort Dynamic Tables
- VBScript: Using object-level events

This section also briefly describes some operations that can be performed within Authentic. This is to give you an idea of what is possible with the XMLSpy API.

4.2.1 JScript: Bubble Sort Dynamic Tables

The following JScript snippet will sort any dynamic table by the table column identified by the current cursor position. The sort process is performed on screen. The undo buffer is available for all performed operations.

If you can run JScript on you computer - as you most likely will - copy the following code into a file with extension 'js'. Execute the script by double-clicking it in Windows Explorer, or run it from the command line.

```
// some useful XMLSpy enum constants
var spyAuthenticTag = 6;
var spyAuthenticTable = 9;
var spyAuthenticTableRow = 10;
var spyAuthenticTableColumn = 11;

var spyAuthenticRangeBegin = 2;

// example call for the sort table function
try
{
  var objSpy = objSpy = WScript.GetObject("", "XMLSpy.Application");
  // use current selection to indicate which column to sort
  SortCurrentTable (objSpy.ActiveDocument.AuthenticView.Selection);
}
catch (err)
  { WScript.Echo ("Please open a document in authentic view, and select a
  table column\n" +
                  "Error : (" + (err.number & 0xffff) + ")" +
err.description); }

// we assume that XMLSpy is running, a document with a dynamic table
// is open, and the cursor is in a table column that will
// be used for sorting.
function SortCurrentTable (objCursor)
{
  if (objCursor.IsInDynamicTable())
  {
    // calculate current column index
    var nColIndex = 0;
    while (true)
    {
```

```
      // go left column-by-column
      try { objCursor.GotoPrevious(spyAuthenticTableColumn); }
      catch (err) { break; }
      nColIndex++;
    }

    // count number of table rows, so the bubble loops become simpler.
    // goto begin of table
    var objTableStart =
    objCursor.ExpandTo(spyAuthenticTable).CollapsToBegin().Clone();
    var nRows = 1;
    while (true)
    {
      // go down row-by-row
      try { objTableStart.GotoNext(spyAuthenticTableRow); }
      catch (err) { break; }
      nRows++;
    }

    // bubble sort through table
    for (var i = 0; i < nRows - 1; i++)
    {
      // select correct column in first table row
      var objBubble =
      objCursor.ExpandTo(spyAuthenticTable).CollapsToBegin().Clone();
      objBubble.Goto (spyAuthenticTableColumn, nColIndex,
      spyAuthenticRangeBegin).ExpandTo(spyAuthenticTag);

      // bubble this row down as far as necessary
      for (var j = 0; j < nRows - i - 1; j++)
      {
        var strField1 = objBubble.Text;
        // now look for the comparison table cell: start of next row and right
        of the correct column
        var strField2 = objBubble.GotoNext(spyAuthenticTableRow).
                                  Goto (spyAuthenticTableColumn, nColIndex,
        spyAuthenticRangeBegin).
                                  ExpandTo(spyAuthenticTag).Text;
        if (strField1 > strField2)
        {
          objBubble.MoveRowUp();  // swap the rows
          // and re-calculate objBubble to select the cell to bubble
          objBubble.GotoNext(spyAuthenticTableRow).
                    Goto (spyAuthenticTableColumn, nColIndex,
          spyAuthenticRangeBegin).
                    ExpandTo(spyAuthenticTag);
        }
      }
    }
  }
  else
    WScript.Echo ("please, select a table cell first");
}
```

4.2.2 VBScript: Using object-level events

Authentic view now 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 VB code snippet that shows how to connect to object-level events from within a VB project.

```
'
```

```
'-------------------------------------------------------------------------
' VB code snippet - connecting to object level events
'
'-------------------------------------------------------------------------
Dim objSpy As XMLSpyLib.Application
' use VBA keyword WithEvents to automatically connect to object-level event
Dim WithEvents objView As XMLSpyLib.AuthenticView

' this is the event callback routine that will be 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

' this is the event callback routine that will be 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

' this is the event callback routine that will be connected to the
' OnMouseEvent event of object objView.
' If you click with the left mouse button while pressing a control key,
' the current selection will be set to the tag below the current
' mouse cursor position
Private Function objView_OnMouseEvent(ByVal i_nXPos As Long, ByVal i_nYPos
As Long, ByVal i_eMouseEvent As XMLSpyLib.SPYMouseEvent, ByVal i_pRange As
XMLSpyLib.IAuthenticRange) As Boolean
    If (i_eMouseEvent = (XMLSpyLib.spyLeftButtonDownMask Or
XMLSpyLib.spyCtrlKeyDownMask)) Then
        On Error Resume Next
        i_pRange.Select
        objView_OnMouseEvent = True
    Else
        objView_OnMouseEvent = False
    End If
End Function

' connect to running XMLSpy application
' this code will most likely be in some Form_Load() subroutine
Set objSpy = GetObject("", "XMLSpy.Application")
objSpy.ShowApplication (True)
If (objSpy.ActiveDocument Is Nothing) Then
    Dim objDoc As XMLSpyLib.Document
    Rem replace path below With a valid absolute path On your machine
    Set objDoc = objSpy.Documents.OpenFile("InputData\orgchart.xml", False)
End If

If (objSpy.ActiveDocument.CurrentViewMode <> spyViewContent) Then
    objSpy.ActiveDocument.SwitchViewMode (spyViewContent)
End If
Set objView = objSpy.ActiveDocument.AuthenticView
' now the object objView is set and selection change events are received

' continue here with something useful ...
' and serve the windows message loop

' if you want to stop receiving events from object objView
' add the following line somewhere appropriate
Set objView = Nothing
```

4.3 Interfaces

Object Hierarchy

Application
- SpyProject
  - SpyProjectItems
    - SpyProjectItem
- Documents
  - Document
    - GridView
    - AuthenticView
      - AuthenticRange
      - AuthenticDataTransfer (previously DocEditDataTransfer)
    - OldAuthenticView (previously DocEditView, **now obsolete**, superseded by AuthenticView and AuthenticRange)
      - AuthenticSelection (previously DocEditSelection, **now obsolete**, superseded by AuthenticRange)
      - AuthenticEvent (previously DocEditEvent, **now obsolete**)
      - AuthenticDataTransfer (previously DocEditDataTransfer)
    - XMLData
- Dialogs
  - CodeGeneratorDlg
  - FileSelectionDlg
  - SchemaDocumentationDlg
- DatabaseConnection
- ExportSettings
- TextImportExportSettings
- ElementList
  - ElementListItem

Enumerations

Description

This chapter contains the reference of the XMLSpy 1.5 Type Library.

Most of the given examples are written in VisualBasic. These code snippets assume that there is a variable defined and set, called **objSpy of type Application**. There are also some code samples written in JavaScript.

4.3.1 Application

See also

Methods

GetDatabaseImportElementList
GetDatabaseSettings
GetDatabaseTables
ImportFromDatabase

GetTextImportElementList
GetTextImportExportSettings
ImportFromText

ImportFromWord

ImportFromSchema

GetExportSettings

© 2007 Altova GmbH

NewProject
OpenProject

AddMacroMenuItem
ClearMacroMenu

ShowForm

ShowApplication

URLDelete
URLMakeDirectory

FindFiles

Quit

Properties
Application
Parent

ActiveDocument
Documents

CurrentProject

Dialogs

WarningNumber
WarningText

Description
Application is the root for all other objects. It is the only object you can create by CreateObject (VisualBasic) or other similar COM related functions.

Example

```
Dim objSpy As Application
Set objSpy = CreateObject("XMLSpy.Application")
```

Events

OnBeforeOpenDocument

See also

Event: OnBeforeOpenDocument(*objDialog* as FileSelectionDlg)

Description
This event gets fired whenever a document gets opened via the OpenFile or OpenURL menu command. It is sent after a document file has been selected but before the document gets opened. The file selection dialog object is initialized with the name of the selected document file. You can modify this selection. To continue the opening of the document leave the FileSelectionDlgDialogAction property of *io_objDialog* at its default value spyDialogOK. To abort the opening of the document set this property to spyDialogCancel.

Examples
Given below are examples of how this event can be scripted.

XMLSpy scripting environment - VBScript:
```
Function On_BeforeOpenDocument(objDialog)
End Function
```

XMLSpy scripting environment - JScript:
```
function On_BeforeOpenDocument(objDialog)
{
}
```

XMLSpy IDE Plugin:
```
IXMLSpyPlugIn.OnEvent (26, ...)  //nEventID=26
```

OnBeforeOpenProject

See also

Event: `OnBeforeOpenProject(objDialog as FileSelectionDlg)`

Description
This event gets fired after a project file has been selected but before the project gets opened. The file selection dialog object is initialized with the name of the selected project file. You can modify this selection. To continue the opening of the project leave the FileSelectionDlg.DialogAction property of *io_objDialog* at its default value spyDialogOK. To abort the opening of the project set this property to spyDialogCancel.

Examples
Given below are examples of how this event can be scripted.

XMLSpy scripting environment - VBScript:
```
Function On_BeforeOpenProject(objDialog)
End Function
```

XMLSpy scripting environment - JScript:
```
function On_BeforeOpenProject(objDialog)
{
}
```

XMLSpy IDE Plugin:
```
IXMLSpyPlugIn.OnEvent (25, ...)  //nEventID=25
```

OnDocumentOpened

See also

Event: `OnDocumentOpened(objDocument as Document)`

Description
This event gets fired whenever a document opens in XMLSpy. This can happen due to opening a file with the OpenFile or OpenURL dialog, creating a new file or dropping a file onto XMLSpy. The new document gets passed as parameter. The operation cannot be canceled.

 © 2007 Altova GmbH

Examples
Given below are examples of how this event can be scripted.

XMLSpy scripting environment - VBScript:
```
Function On_OpenDocument(objDocument)
End Function
```

XMLSpy scripting environment - JScript:
```
function On_OpenDocument(objDocument)
{
}
```

XMLSpy IDE Plugin:
```
IXMLSpyPlugIn.OnEvent (7, ...)    //nEventID=7
```

OnProjectOpened

See also

Event: `OnProjectOpened(objProject as SpyProject)`

Description
This event gets fired whenever a project gets opened in XMLSpy. The new project gets passed as parameter.

Examples
Given below are examples of how this event can be scripted.

XMLSpy scripting environment - VBScript:
```
Function On_OpenProject(objProject)
End Function
```

XMLSpy scripting environment - JScript:
```
function On_OpenProject(objProject)
{
}
```

XMLSpy IDE Plugin:
```
IXMLSpyPlugIn.OnEvent (6, ...)    //nEventID=6
```

ActiveDocument

See also

Property: `ActiveDocument` as `Document`

Description
Reference to the active document. If no document is open, `ActiveDocument` is null (nothing).

Errors
1111 The application object is no longer valid.

1100 Invalid address for the return parameter was specified.

AddMacroMenuItem

See also

Method: `AddMacroMenuItem(strMacro as String,strDisplayText as String)`

Return Value

Description
Adds an menu item to the **Tools** menu. This new menu item invokes the macro defined by strMacro . See also Calling macros from XMLSpy".

Errors
1111 The application object is no longer valid.
1100 Invalid parameter or invalid address for the return parameter was specified.
1108 Number of macro items is limited with 16 items.

Application

See also

Property: `Application` as `Application` (read-only)

Description
Access the XMLSpy application object.

Errors
1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

ClearMacroMenu

See also

Method: `ClearMacroMenu( )`

Return Value
None

Description
Removes all menu items from the **Tools** menu. See also Calling macros from XMLSpy".

Errors
1111 The application object is no longer valid.

CurrentProject

See also

Property: `CurrentProject` as `SpyProject`

© 2007 Altova GmbH

Description
Reference to the active document. If no project is open, CurrentProject is null (nothing).

Errors
1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

Dialogs

See also

Property: Dialogs as Dialogs (read-only)

Description
Access the built-in dialogs of XMLSpy.

Errors
1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

Documents

See also

Property: Documents as Documents

Description
Collection of all open documents. See also Simple document access.

Errors
1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

FindInFiles

See also

Method: FindInFiles(*pSettings* as FindInFilesDlg) as FindInFilesResults

Description
Returns a FindInFilesResults object containing information about the files that matched the specified settings.

Errors
1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

GetDatabaseImportElementList

See also

Method: GetDatabaseImportElementList(*pImportSettings* as DatabaseConnection) as ElementList

Description
The function returns a collection of ElementListItems where the properties ElementListItem.Name contain the names of the fields that can be selected for import and the properties ElementListItem.ElementKind are initialized either to *spyXMLDataAttr* or *spyXMLDataElement*, depending on the value passed in DatabaseConnection.AsAttributes. This list serves as a filter to what finally gets imported by a future call to ImportFromDatabase. Use ElementList.RemoveElement to exclude fields from import.

Properties mandatory to be filled out for the database connection are one of DatabaseConnection.File, DatabaseConnection.ADOConnection and DatabaseConnection.ODBCConnection, as well as DatabaseConnection.SQLSelect. Use the property DatabaseConnection.AsAttributes to initialize ElementListItem.ElementKind of the resulting element list to either *spyXMLDataAttr* or *spyXMLDataElement*, respectively.

Example
See example at ImportFromDatabase.

Errors
1111 The application object is no longer valid.
1100 Invalid parameter or invalid address for the return parameter was specified.
1107 Import from database failed.
1112 Invalid database specified.
1114 Select statement is missing.
1119 database element list import failed.

GetDatabaseSettings

See also

Method: GetDatabaseSettings() as DatabaseConnection

Description
GetDatabaseSettings creates a new object of database settings. The object is used to specify database connection parameters for the methods GetDatabaseTables, GetDatabaseImportElementList, ImportFromDatabase, ImportFromSchema and ExportToDatabase.

Example
See example of ImportFromDatabase.

Errors
1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

GetDatabaseTables

See also

Method: GetDatabaseTables(*pImportSettings* as DatabaseConnection) as ElementList

Description
GetDatabaseTables reads the table names from the database specified in *pImportSettings*.

 © 2007 Altova GmbH

Properties mandatory to be filled out for the database connection are one of DatabaseConnection.File, DatabaseConnection.ADOConnection and DatabaseConnection.ODBCConnection. All other properties are ignored. The function returns a collection of ElementListItems where the properties ElementListItem.Name contain the names of tables stored in the specified database. The remaining properties of ElementListItem are unused.

Errors

1111 The application object is no longer valid.
1100 Invalid parameter or invalid address for the return parameter was specified.
1112 Invalid database specified.
1113 Error while reading database table information.
1118 Database table query failed.

Example

```
Dim objImpSettings As DatabaseConnection
Set objImpSettings = objSpy.GetDatabaseSettings
objImpSettings.ADOConnection = TxtADO.Text

'store table names in 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
```

GetExportSettings

See also

Method: GetExportSettings() as ExportSettings (read-only)

Description

GetExportSettings creates a new object of common export settings. This object is used to pass the parameters to the export functions and defines the behaviour of the export calls. See also the export functions from Document and the examples at Import and Export.

Errors

1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

GetTextImportElementList

See also

Method: GetTextImportElementList(pImportSettings as TextImportExportSettings) as ElementList

Description

GetTextImportElementList retrieves importing information about the text-file as specified in pImportSettings. The function returns a collection of ElementListItems where the properties ElementListItem.Name contain the names of the fields found in the file. The values of

remaining properties are undefined.

If the text-file does not contain a column header, set pImportSettings.HeaderRow to false. The resulting element list will contain general column names like 'Field1' and so on.

See also Import and export of data.

Errors

1111 The application object is no longer valid.
1100 Invalid parameter or invalid address for the return parameter was specified.
1107 Import from database failed.
1115 Error during text element list import. Cannot create parser for import file.
1116 Error during text element list import.

Example

```
' ----------------------------------------------------------
' VBA client code fragment - import selected fields from text file
' ----------------------------------------------------------
Dim objImpSettings As TextImportExportSettings
Set objImpSettings = objSpy.GetTextImportExportSettings

objImpSettings.ImportFile = "C:\ImportMe.txt"
objImpSettings.HeaderRow = False

Dim objList As ElementList
Set objList = objSpy.GetTextImportElementList(objImpSettings)

'exclude first column
objList.RemoveItem 1

Dim objImpDoc As Document
On Error Resume Next
Set objImpDoc = objSpy.ImportFromText(objImpSettings, objList)
CheckForError
```

GetTextImportExportSettings

See also

Method: GetTextImportExportSettings() as TextImportExportSettings (read-only)

Description

GetTextImportExportSettings creates a new object of common import and export settings for text files. See also the example for Application.GetTextImportElementList and Import and Export.

See also Import and export of data.

Errors

1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

ImportFromDatabase

See also

Method: ImportFromDatabase(pImportSettings as DatabaseConnection

© 2007 Altova GmbH

pElementList as ElementList) as Document

Return Value
Creates a new document containing the data imported from the database.

Description
ImportFromDatabase imports data from a database as specified in pImportSettings and creates a new document containing the data imported from the database. Properties mandatory to be filled out are one of DatabaseConnection.File, DatabaseConnection.ADOConnection or DatabaseConnection.ODBCConnection and DatabaseConnection.SQLSelect. Additionally, you can use DatabaseConnection.AsAttributes, DatabaseConnection.ExcludeKeys, DatabaseConnection.IncludeEmptyElements and NumberDateTimeFormat to further parameterize import.

The parameter pElementList specifies which fields of the selected data gets written into the newly created document, and which are created as elements and which as attributes. This parameter can be NULL, specifying that all selected fields will be imported as XML elements.

See GetDatabaseSettings and GetDatabaseImportElementList for necessary steps preceding any import of data from a database.

Errors

1111	The application object is no longer valid.
1100	Invalid parameter or invalid address for the return parameter was specified.
1107	Import from database failed.
1112	Invalid database specified.
1114	Select statement is missing.
1117	Transformation to XML failed.
1120	Database import failed.

Example

```
 Dim objImpSettings As DatabaseConnection
 Set objImpSettings = objSpy.GetDatabaseSettings

 objImpSettings.ADOConnection = strADOConnection
 objImpSettings.SQLSelect = "SELECT * FROM MyTable"

 Dim objDoc As Document
 On Error Resume Next
 Set objDoc = objSpy.ImportFromDatabase(objImpSettings,
objSpy.GetDatabaseImportElementList(objImpSettings))
 ' CheckForError here
```

ImportFromSchema

See also

Method: ImportFromSchema(*pImportSettings* as DatabaseConnection,*strTable* as String,*pSchemaDoc* as Document) as Document

Return Value
Creates a new document filled with data from the specified database as specified by the schema definition in *pSchemaDoc*.

Description

ImportFromSchema imports data from a database specified in pImportSettings. Properties mandatory to be filled out are one of DatabaseConnection.File, DatabaseConnection.ADOConnection or DatabaseConnection.ODBCConnection. Additionally, you can use DatabaseConnection.AsAttributes, DatabaseConnection.ExcludeKeys and NumberDateTimeFormat to further parameterize import. All other properties get ignored.

ImportFromSchema does not use and explicit SQL statement to select the data. Instead, it expects a structure definition of the document to create in form of an XML schema document in *pSchemaDoc*. From this definition the database select statement is automatically deduced. Specify in *strTable* the table name of the import root that will become the root node in the new document.

See GetDatabaseSettings and GetDatabaseTables for necessary steps preceding an import from a database based on a schema definition. To create the schema definition file use command 'create database schema' from the 'convert' menu of XMLSpy.

Errors

1111	The application object is no longer valid.
1100	Invalid parameter or invalid address for the return parameter was specified.
1107	Import from database failed.
1112	Invalid database specified.
1120	Database import failed.
1121	Could not create validator for the specified schema.
1122	Failed parsing schema for database import.

ImportFromText

See also

Method: ImportFromText(*pImportSettings* as TextImportExportSettings, *pElementList* as ElementList) as Document

Description

ImportFromText imports the text file as specified in pImportSettings. The parameter pElementList can be used as import filter. Either pass the list returned by a previous call to GetTextImportElementList or nil to import all columns. To avoid import of unnecessary columns use ElementList.RemoveElement to remove the corresponding field names from pElementList before calling ImportFromText.
The method returns the newly created document containing the imported data. This document is the same as the active document of XMLSpy.

See also Import and export of data.

Errors

1111	The application object is no longer valid.
1100	Invalid parameter or invalid address for the return parameter was specified.
1107	Import from text file failed.
1117	Transformation to XML failed.

Example

```
' ------------------------------------------------------
' VBA client code fragment - import from text file
' ------------------------------------------------------
Dim objImpSettings As TextImportExportSettings
```

 © 2007 Altova GmbH

```
Set objImpSettings = objSpy.GetTextImportExportSettings

objImpSettings.ImportFile = strFileName
objImpSettings.HeaderRow = False

Dim objImpDoc As Document
On Error Resume Next
Set objImpDoc = objSpy.ImportFromText(objImpSettings,
  objSpy.GetTextImportElementList(objImpSettings))

CheckForError
```

ImportFromWord

See also

Method: ImportFromWord(*strFile* as String) as Document

Description

ImportFromWord imports the MS-Word Document strFile into a new XML document.

Errors

1111 The application object is no longer valid.
1100 Invalid parameter or invalid address for the return parameter was specified.
Import from document failed.

NewProject

See also

Method: NewProject(*strPath* as String,*bDiscardCurrent* as Boolean)

Description

NewProject creates a new project.

If there is already a project open that has been modified and bDiscardCurrent is false, then NewProject() fails.

Errors

1111 The application object is no longer valid.
1102 A project is already open but *bDiscardCurrent* is *true*.
1103 Creation of new project failed.

OpenProject

See also

Method: OpenProject(*strPath* as String,*bDiscardCurrent* as Boolean,*bDialog* as Boolean)

Parameters

strPath
Path and file name of the project to open. Can be empty if bDialog is true.

bDiscardCurrent
Discard currently open project and possible lose changes.

bDialog
Show dialogs for user input.

Return Value
None

Description
OpenProject opens an existing project. If there is already a project open that has been modified and bDiscardCurrent is false, then OpenProject() fails.

Errors
1111 The application object is no longer valid.
1100 Invalid parameter or invalid address for the return parameter was specified.
1101 Cannot open specified project.
1102 A project is already open but *bDiscardCurrent* is *true*.

Parent

See also

Property: Parent as Application (read-only)

Description
Access the XMLSpy application object.

Errors
1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

Quit

See also

Method: Quit()

Return Value
None

Description
This method terminates XMLSpy. All modified documents will be closed without saving the changes. This is also true for an open project.

If XMLSpy was automatically started as an automation server by a client program, the application will not shut down automatically when your client program shuts down if a project or any document is still open. Use the Quit method to ensure automatic shut-down.

Errors
1111 The application object is no longer valid.

ReloadSettings

See also

Method: ReloadSettings

 © 2007 Altova GmbH

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.

RunMacro

See also

Method: `RunMacro(strMacro as String)`

Return Value

Description
Calls the specified macro either from the project scripts (if present) or from the global scripts.

Available with TypeLibrary version 1.5

Errors
1111 The application object is no longer valid.

ScriptingEnvironment

See also

Property: `ScriptingEnvironment` as IUnknown (read-only)

Description
Reference to any active scripting environment. This property makes it possible to access the TypeLibrary of the XMLSpyFormEditor.exe application which is used as the current scripting environment.

Available with TypeLibrary version 1.5

Errors
1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

ShowApplication

See also

Method: `ShowApplication( bShow as Boolean)`

Return Value
None

Description
The method shows (`bShow = True`) or hides (`bShow = False`) XMLSpy.

Errors
1110 The application object is no longer valid.

ShowFindInFiles

See also

Method: ShowFindInFiles(pSettings as FindInFilesDlg) as Boolean

Return Value
Returns false if the user pressed the Cancel button, true otherwise.

Description
Displays the FindInFiles dialog preset with the given settings. The user modifications of the settings are stored in the passed dialog object.

Errors
1111 The application object is no longer valid.
1100 Invalid parameter or invalid address for the return parameter was specified.

ShowForm

See also

Method: ShowForm(*strFormName* as String) as Long

Return Value
Returns zero if the user pressed a Cancel button or the form calls TheView.Cancel() .

Description
Displays the form strFormName .

Forms, event handlers and macros can be created with the Scripting Environment. Select "Switch to scripting environment" from the **Tools** menu to invoke the Scripting Environment.

Errors
1111 The application object is no longer valid.
1100 Invalid parameter or invalid address for the return parameter was specified.

URLDelete

See also

Method: URLDelete(*strURL* as String,*strUser* as String,*strPassword* as String)

Return Value
None

Description
The method deletes the file at the URL strURL .

Errors
1111 The application object is no longer valid.
1109 Error deleting file at specified URL.

© 2007 Altova GmbH

URLMakeDirectory

See also

Method: URLMakeDirectory(*strURL* as String,*strUser* as String,*strPassword* as String)

Return Value
None

Description
The method creates a new directory at the URL strURL .

Errors
1111 The application object is no longer valid.
1100 Invalid parameter specified.

Visible

See also

Property: Visible as Boolean

Description
Sets or gets the visibility attribute of XMLSpy. This standard automation property makes usage of ShowApplication obsolete.

Errors
1110 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

WarningNumber

See also

Property: WarningNumber as integer

Description
Some methods fill the property WarningNumber with additional information if an error occurs.

Currently just Documents.OpenFile fills this property.

Errors
1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

WarningText

See also

Property: WarningText as String

Description
Some methods fill the property WarningText with additional information if an error occurs.

Currently just Documents.OpenFile fills this property.

Errors

1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

4.3.2 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 interface. All functionality available up to now in DocEditView, DocEditSelection, DocEditEvent and DocEditDataTransfer is now available via AuthenticView, AuthenticRange and AuthenticDataTransfer. Many new features have been added.

For examples on migrating from DocEdit to Authentic see the description of the different methods and properties of the different DocEdit objects.

See also

Methods

getData

Properties

dropEffect
ownDrag
type

Description

The events OnDragOver and OnBeforeDrop provide information about the object being dragged with an instance of type AuthenticDataTransfer. It contains a description of the dragged object and its content. The latter is available either as string or a pointer to a COM object supporting the IUnkown interface.

dropEffect

See also

Property: dropEffect as long

Description
The property stores the drop effect from the default event handler. You can set the drop effect if you change this value and return TRUE for the event handler (or set AuthenticEvent.cancelBubble to TRUE if you are still using the now obsolete

 © 2007 Altova GmbH

AuthenticEvent interface).

Errors
2101 Invalid address for the return parameter was specified.

getData

See also

Method: `getData()` as Variant

Description
Retrieve the data associated with the dragged object. Depending on `AuthenticDataTransfer.type`, that data is either a string or a COM interface pointer of type `IUnknown`.

Errors
2101 Invalid address for the return parameter was specified.

ownDrag

See also

Property: `ownDrag` as Boolean (read-only)

Description
The property is `TRUE` if the current dragging source comes from inside Authentic.

Errors
2101 Invalid address for the return parameter was specified.

type

See also

Property: `type` as String (read-only)

Description
Holds the type of data you get with the `DocEditDataTransfer.getData` method.

Currently supported data types are:

OWN	data from Authentic itself
TEXT	plain text
UNICODETEXT	plain text as UNICODE

Errors
2101 Invalid address for the return parameter was specified.

4.3.3 AuthenticRange

See also

The first table lists the properties and methods of AuthenticRange that can be used to navigate through the document and select specific portions.

Properties

Application
FirstTextPosition
FirstXMLData
FirstXMLDataOffset
LastTextPosition
LastXMLData
LastXMLDataOffset
Parent

Clone
CollapsToBegin
CollapsToEnd
ExpandTo
Goto
GotoNext
GotoPrevious
IsEmpty
IsEqual

Methods

MoveBegin
MoveEnd
NextCursorPosition
PreviousCursorPosition
Select
SelectNext
SelectPrevious
SetFromRange

The following table lists the content modification methods, most of which can be found on the right/button mouse menu.

Properties	Edit operations	Dynamic table operations
Text	Copy	AppendRow
	Cut	DeleteRow
	Delete	DuplicateRow
	IsCopyEnabled	InsertRow
	IsCutEnabled	IsFirstRow
	IsDeleteEnabled	IsInDynamicTable
	IsPasteEnabled	IsLastRow
	Paste	MoveRowDown
		MoveRowUp

The following methods provide the functionality of the Authentic entry helper windows for range objects.

Operations of the entry helper windows

Elements	Attributes	Entities
CanPerformActionWith	GetElementAttributeValue	GetEntityNames
CanPerformAction	GetElementAttributeNames	InsertEntity
PerformAction	GetElementHierarchy	
	HasElementAttribute	
	IsTextStateApplied	
	SetElementAttributeValue	

Description

AuthenticRange objects are the 'cursor' selections of the automation interface. You can use them to point to any cursor position in the Authentic view, or select a portion of the document. The operations available for AuthenticRange objects then work on this selection in the same way, as the corresponding operations of the user interface do with the current user interface selection. The main difference is that you can use an arbitrary number of AuthenticRange objects at the same time, whereas there is exactly one cursor selection in the user interface.

To get to an initial range object use AuthenticView.Selection, to obtain a range corresponding with the current cursor selection in the user interface. Alternatively, some trivial ranges are accessible via the read/only properties AuthenticView.DocumentBegin, AuthenticView.DocumentEnd, and AuthenticView.WholeDocument. The most flexible method is AuthenticView.Goto, which allows navigation to a specific portion of the document within one call. For more complex selections, combine the above, with the various navigation methods on range objects listed in the first table on this page.

Another method to select a portion of the document is to use the position properties of the range object. Two positioning systems are available and can be combined arbitrarily:

© 2007 Altova GmbH

- **Absolute** text cursor positions, starting with position 0 at the document beginning, can be set and retrieved for the beginning and end of a range. For more information see FirstTextPosition and LastTextPosition. This method requires complex internal calculations and should be used with care.
- The **XMLData** element and a text position inside this element, can be set and retrieved for the beginning and end of a range. For more information see FirstXMLData, FirstXMLDataOffset, LastXMLData, and LastXMLDataOffset. This method is very efficient but requires knowledge on the underlying document structure. It can be used to locate XMLData objects and perform operations on them otherwise not accessible through the user interface.

Modifications to the document content can be achieved by various methods:

- The Text property allows you to retrieve the document text selected by the range object. If set, the selected document text gets replaced with the new text.
- The standard document edit functions Cut, Copy, Paste and Delete.
- Table operations for tables that can grow dynamically.
- Methods that map the functionality of the Authentic entry helper windows.
- Access to the XMLData objects of the underlying document to modify them directly.

AppendRow

See also

Method: AppendRow( ) as Boolean

Description
If the beginning of the range is inside a dynamic table, this method inserts a new row at the end of the selected table. The selection of the range is modified to point to the beginning of the new row. The function returns *true* if the append operation was successful, otherwise *false*.

Errors

2001	The authentic range object or its related view object is no longer valid.
2005	Invalid address for the return parameter was specified.

Examples

```
' ------------------------------------------------------
' XMLSpy scripting environment - VBScript
' Append row at end of current dynamically growable table
' ------------------------------------------------------
Dim objRange
' we assume that the active document is open in authentic view mode
Set objRange = Application.ActiveDocument.AuthenticView.Selection

' check if we can insert something
If objRange.IsInDynamicTable Then
     objRange.AppendRow
     ' objRange points to beginning of new row
     objRange.Select
End If
```

Application

See also

Property: Application as Application (read-only)

Description
Access the XMLSpy application object.

Errors
2001 The authentic range object or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

CanPerformAction

See also

Method: `CanPerformAction` (*`eAction`* as `SPYAuthenticActions`, *`strElementName`* as String) as Boolean

Description
`CanPerformAction` and its related methods enable access to the entry-helper functions of Authentic. This function allows easy and consistent modification of the document content, without having to know exactly where the modification will take place. The beginning of the range object is used to locate the next valid location where the specified action can be performed. If the location can be found, the method returns *True*, otherwise it returns *False*.

HINT: To find out all valid element names for a given action, use `CanPerformActionWith`.

Errors
2001 The authentic range object or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.
2007 Invalid action was specified.

Examples
See `PerformAction`.

CanPerformActionWith

See also

Method: `PerformActionWith` (*`eAction`* as `SPYAuthenticActions`, *`out_arrElementNames`* as Variant)

Description
`PerformActionWith` and its related methods, enable access to the entry-helper functions of Authentic. These function allows easy and consistent modification of the document content without having to know exactly where the modification will take place.

This method returns an array of those element names that the specified action can be performed with.

HINT: To apply the action use `CanPerformActionWith`.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.
2007 Invalid action was specified.

Examples
See `PerformAction`.

© 2007 Altova GmbH

Clone

See also

Method: `Clone()` as `AuthenticRange`

Description
Returns a copy of the range object.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

CollapsToBegin

See also

Method: `CollapsToBegin()` as `AuthenticRange`

Description
Sets the end of the range object to its begin. The method returns the modified range object.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

CollapsToEnd

See also

Method: `CollapsToEnd()` as `AuthenticRange`

Description
Sets the beginning of the range object to its end. The method returns the modified range object.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

Copy

See also

Method: `Copy()` as Boolean

Description
Returns *False* if the range contains no portions of the document that may be copied.
Returns *True* if text, and in case of fully selected XML elements the elements as well, has been copied to the copy/paste buffer.

Errors
2001 The authentic range object or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

Cut

See also

Method: `Cut()` as Boolean

Description
Returns *False* if the range contains portions of the document that may not be deleted. Returns *True* after text, and in case of fully selected XML elements the elements as well, has been deleted from the document and saved in the copy/paste buffer.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

Delete

See also

Method: `Delete()` as Boolean

Description
Returns *False* if the range contains portions of the document that may not be deleted. Returns *True* after text, and in case of fully selected XML elements the elements as well, has been deleted from the document.

Errors
2001 The authentic range object or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

DeleteRow

See also

Method: `DeleteRow()` as Boolean

Description
If the beginning of the range is inside a dynamic table, this method deletes the selected row. The selection of the range gets modified to point to the next element after the deleted row. The function returns *true,* if the delete operation was successful, otherwise *false*.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

Examples

```
' ----------------------------------------------------------
' XMLSpy 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
```

 © 2007 Altova GmbH

```
     objRange.DeleteRow
End If
```

DuplicateRow

See also

Method: DuplicateRow() as Boolean

Description
If the beginning of the range is inside a dynamic table, this method inserts a duplicate of the current row after the selected one. The selection of the range gets modified to point to the beginning of the new row. The function returns *true* if the duplicate operation was successful, otherwise *false*.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

Examples

```
' ---------------------------------------------------------
' XMLSpy 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
```

ExpandTo

See also

Method: ExpandTo (*eKind* as SPYAuthenticElementKind), as AuthenticRange

Description
Selects the whole element of type eKind , that starts at, or contains, the first cursor position of the range. The method returns the modified range object.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2003 Range expansion would be beyond end of document.
2005 Invalid address for the return parameter was specified.

FirstTextPosition

See also

Property: FirstTextPosition as Long

Description
Set or get the left-most text position index of the range object. This index is always less or equal

to LastTextPosition. Indexing starts with 0 at document beginning, and increments with every different position that the text cursor can occupy. Incrementing the test position by 1, has the same effect as the cursor-right key. Decrementing the test position by 1 has the same effect as the cursor-left key.

If you set FirstTextPosition to a value greater than the current LastTextPosition, LastTextPosition gets set to the new FirstTextPosition.

HINT: Use text cursor positions with care, since this is a costly operation compared to XMLData based cursor positioning.

Errors

2001 The authentic range object, or its related view object is not valid.
2005 Invalid address for the return parameter was specified.
2006 A text position outside the document was specified.

Examples

```
' ---------------------------------------
' XMLSpy scripting environment - VBScript
' ---------------------------------------
Dim objAuthenticView
' we assume that the active document is open in authentic view mode
Set objAuthenticView = Application.ActiveDocument.AuthenticView

nDocStartPosition = objAuthenticView.DocumentBegin.FirstTextPosition
nDocEndPosition = objAuthenticView.DocumentEnd.FirstTextPosition

' let's create a range that selects the whole document
' in an inefficient way
Dim objRange
' we need to get a (any) range object first
Set objRange = objAuthenticView.DocumentBegin
objRange.FirstTextPosition = nDocStartPosition
objRange.LastTextPosition = nDocEndPosition

' let's check if we got it right
If objRange.isEqual(objAuthenticView.WholeDocument) Then
     MsgBox "Test using direct text cursor positioning was ok"
Else
     MsgBox "Ooops!"
End If
```

FirstXMLData

See also

Property: FirstXMLData as XMLData

Description

Set or get the first XMLData element in the underlying document that is partially, or completely selected by the range. The exact beginning of the selection is defined by the FirstXMLDataOffset attribute.

Whenever you set FirstXMLData to a new data object, FirstXMLDataOffset gets set to the first cursor position inside this element. Only XMLData objects that have a cursor position may be used. If you set FirstXMLData / FirstXMLDataOffset selects a position greater then the current LastXMLData / LastXMLDataOffset, the latter gets moved to the new start position.

HINT: You can use the FirstXMLData and LastXMLData properties, to directly access and manipulate the underlying XML document in those cases where the methods available with the

 © 2007 Altova GmbH

AuthenticRange object are not sufficient.

Errors

2001 The authentic range object, or its related view object is not valid.
2005 Invalid address for the return parameter was specified.
2008 Internal error
2009 The XMLData object cannot be accessed.

Examples

```
' ------------------------------------------
' XMLSpy scripting environment - VBScript
' show name of currently selected XMLData element
' ------------------------------------------
Dim objAuthenticView
' we assume that the active document is open in authentic view mode
Set objAuthenticView = Application.ActiveDocument.AuthenticView

Dim objXmlData
Set objXMLData = objAuthenticView.Selection.FirstXMLData
' authentic view adds a 'text' child element to elements
' of the document which have content. So we have to go one
' element up.
Set objXMLData = objXMLData.Parent
MsgBox "Current selection selects element " & objXMLData.Name
```

FirstXMLDataOffset

See also

Property: FirstXMLDataOffset as Long

Description

Set or get the cursor position offset inside FirstXMLData element for the beginning of the range. Offset positions are based on the characters returned by the Text property, and start with 0. When setting a new offset, use -1 to set the offset to the last possible position in the element. The following cases require specific attention:

- The textual form of entries in Combo Boxes, Check Boxes and similar controls can be different from what you see on screen. Although the data offset is based on this text, there only two valid offset positions, one at the beginning and one at the end of the entry. An attempt to set the offset to somewhere in the middle of the entry, will result in the offset being set to the end.
- The textual form of XML Entities might differ in length from their representation on the screen. The offset is based on this textual form.

If FirstXMLData / FirstXMLDataOffset selects a position after the current LastXMLData / LastXMLDataOffset, the latter gets moved to the new start position.

Errors

2001 The authentic range object, or its related view object is not valid.
2005 Invalid offset was specified.
Invalid address for the return parameter was specified.

Examples

```
' ------------------------------------------
' XMLSpy scripting environment - VBScript
' Select the complete text of an XMLData element
' using XMLData based selection and ExpandTo
' ------------------------------------------
```

```
Dim objAuthenticView
' we assume that the active document is open in authentic view mode
Set objAuthenticView = Application.ActiveDocument.AuthenticView

' first we use the XMLData based range properties
' to select all text of the first XMLData element
' in the current selection
Dim objRange
Set objRange = objAuthenticView.Selection
objRange.FirstXMLDataOffset = 0  ' start at beginning of element text
objRange.LastXMLData = objRange.FirstXMLData  ' select only one element
objRange.LastXMLDataOffset = -1  ' select till its end

' the same can be achieved with the ExpandTo method
Dim objRange2
Set objRange2 = objAuthenticView.Selection.ExpandTo(spyAuthenticTag)

' were we successful?
If objRange.IsEqual(objRange2) Then
     objRange.Select()
Else
     MsgBox "Oops"
End If
```

GetElementAttributeNames

See also

Method: GetElementAttributeNames (*strElementName* as String, *out_arrAttributeNames* as Variant)

Description
Retrieve the names of all attributes for the enclosing element with the specified name. Use the element/attribute pairs, to set or get the attribute value with the methods GetElementAttributeValue and SetElementAttributeValue.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid element name was specified.
Invalid address for the return parameter was specified.

Examples
See SetElementAttributeValue.

GetElementAttributeValue

See also

Method: GetElementAttributeValue (*strElementName* as String, *strAttributeName* as String) as String

Description
Retrieve the value of the attribute specified in strAttributeName, for the element identified with strElementName. If the attribute is supported but has no value assigned, the empty string is returned. To find out the names of attributes supported by an element, use GetElementAttributeNames, or HasElementAttribute.

Errors
2001 The authentic range object, or its related view object is no longer valid.

 © 2007 Altova GmbH

2005 Invalid element name was specified.
Invalid attribute name was specified.
Invalid address for the return parameter was specified.

Examples
See SetElementAttributeValue.

GetElementHierarchy

See also

Method: GetElementHierarchy (*out_arrElementNames* as Variant)

Description
Retrieve the names of all XML elements that are parents of the current selection. Inner elements get listed before enclosing elements. An empty list is returned whenever the current selection is not inside a single XMLData element.

The names of the element hierarchy, together with the range object uniquely identify XMLData elements in the document. The attributes of these elements can be directly accessed by GetElementAttributeNames, and related methods.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

Examples
See SetElementAttributeValue.

GetEntityNames

See also

Method: GetEntityNames (*out_arrEntityNames* as Variant)

Description
Retrieve the names of all defined entities. The list of retrieved entities is independent of the current selection, or location. Use one of these names with the InsertEntity function.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

Examples
See InsertEntity.

Goto

See also

Method: Goto (*eKind* as SPYAuthenticElementKind, *nCount* as Long, *eFrom* as SPYAuthenticDocumentPosition) as AuthenticRange

Description
Sets the range to point to the beginning of the nCount element of type eKind . The start position

is defined by the parameter eFrom .

Use positive values for nCount to navigate to the document end. Use negative values to navigate to the beginning of the document. The method returns the modified range object.

Errors

2001	The authentic range object, or its related view object is no longer valid.
2003	Target lies after end of document.
2004	Target lies before begin of document.
2005	Invalid element kind specified. Invalid start position specified. Invalid address for the return parameter was specified.

GotoNext

See also

Method: GotoNext (*eKind* as SPYAuthenticElementKind) as AuthenticRange

Description

Sets the range to the beginning of the next element of type eKind . The method returns the modified range object.

Errors

2001	The authentic range object, or its related view object is no longer valid.
2003	Target lies after end of document.
2005	Invalid element kind specified. Invalid address for the return parameter was specified.

Examples

```
' ------------------------------------------
' XMLSpy scripting environment - VBScript
' Scan through the whole document word-by-word
' ------------------------------------------
Dim objAuthenticView
' we assume that the active document is open in authentic view mode
Set objAuthenticView = Application.ActiveDocument.AuthenticView

Dim objRange
Set objRange = objAuthenticView.DocumentBegin
Dim bEndOfDocument
bEndOfDocument = False

On Error Resume Next
While Not bEndOfDocument
     objRange.GotoNext(spyAuthenticWord).Select
     If ((Err.number - vbObjecterror) = 2003) Then
            bEndOfDocument = True
            Err.Clear
     ElseIf (Err.number <> 0) Then
            Err.Raise  ' forward error
     End If
Wend
```

GotoNextCursorPosition

See also

Method: GotoNextCursorPosition() as AuthenticRange

© 2007 Altova GmbH

Description
Sets the range to the next cursor position after its current end position. Returns the modified object.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2003 Target lies after end of document.
2005 Invalid address for the return parameter was specified.

GotoPrevious

See also

Method: GotoPrevious (*eKind* as SPYAuthenticElementKind) as AuthenticRange

Description
Sets the range to the beginning of the element of type eKind which is before the beginning of the current range. The method returns the modified range object.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2004 Target lies before beginning of document.
2005 Invalid element kind specified.
Invalid address for the return parameter was specified.

Examples

```
' ------------------------------------------
' XMLSpy scripting environment - VBScript
' Scan through the whole document tag-by-tag
' ------------------------------------------
Dim objAuthenticView
' we assume that the active document is open in authentic view mode
Set objAuthenticView = Application.ActiveDocument.AuthenticView

Dim objRange
Set objRange = objAuthenticView.DocumentEnd
Dim bEndOfDocument
bBeginOfDocument = False

On Error Resume Next
While Not bBeginOfDocument
     objRange.GotoPrevious(spyAuthenticTag).Select
     If ((Err.number - vbObjecterror) = 2004) Then
            bBeginOfDocument = True
            Err.Clear
     ElseIf (Err.number <> 0) Then
            Err.Raise  ' forward error
     End If
Wend
```

GotoPreviousCursorPosition

See also

Method: GotoPreviousCursorPosition() as AuthenticRange

Description
Set the range to the cursor position immediately before the current position. Returns the

modified object.

Errors

2001 The authentic range object, or its related view object is no longer valid.
2004 Target lies before begin of document.
2005 Invalid address for the return parameter was specified.

HasElementAttribute

See also

Method: `HasElementAttribute` (*`strElementName`* as String, *`strAttributeName`* as String) as Boolean

Description

Tests if the enclosing element with name `strElementName`, supports the attribute specified in `strAttributeName`.

Errors

2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid element name was specified.
Invalid address for the return parameter was specified.

InsertEntity

See also

Method: `InsertEntity` (*`strEntityName`* as String)

Description

Replace the ranges selection with the specified entity. The specified entity must be one of the entity names returned by `GetEntityNames`.

Errors

2001 The authentic range object, or its related view object is no longer valid.
2005 Unknown entry name was specified.

Examples

```
' ------------------------------------------------------------
' XMLSpy 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)
     objRange.Select()
Else
     MsgBox "Sorry, no entities are available for this document"
End If
```

 © 2007 Altova GmbH

InsertRow

See also

Method: `InsertRow()` as Boolean

Description
If the beginning of the range is inside a dynamic table, this method inserts a new row before the current one. The selection of the range, gets modified to point to the beginning of the newly inserted row. The function returns *true* if the insert operation was successful, otherwise *false*.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

Examples

```
' ------------------------------------------------------------
' XMLSpy scripting environment - VBScript
' Insert row at beginning of current dynamically growing table
' ------------------------------------------------------------
Dim objRange
' we assume that the active document is open in authentic view mode
Set objRange = Application.ActiveDocument.AuthenticView.Selection

' check if we can insert something
If objRange.IsInDynamicTable Then
     objRange.InsertRow
     ' objRange points to beginning of new row
     objRange.Select
End If
```

IsCopyEnabled

See also

Property: `IsCopyEnabled` as Boolean (read-only)

Description
Checks if the copy operation is supported for this range.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

IsCutEnabled

See also

Property: `IsCutEnabled` as Boolean (read-only)

Description
Checks if the cut operation is supported for this range.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

IsDeleteEnabled

See also

Property: `IsDeleteEnabled` as Boolean (read-only)

Description
Checks if the delete operation is supported for this range.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

IsEmpty

See also

Method: `IsEmpty()` as Boolean

Description
Tests if the first and last position of the range are equal.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

IsEqual

See also

Method: `IsEqual` (*`objCmpRange`* as `AuthenticRange`) as Boolean

Description
Tests if the start and end of both ranges are the same.

Errors
2001 One of the two range objects being compared, is invalid.
2005 Invalid address for a return parameter was specified.

IsFirstRow

See also

Property: `IsFirstRow()` as Boolean (read-only)

Description
Test if the range is in the first row of a table. Which table is taken into consideration depends on the extend of the range. If the selection exceeds a single row of a table, the check is if this table is the first element in an embedding table. See the entry helpers of the user manual for more information.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

© 2007 Altova GmbH

IsInDynamicTable

See also

Method: `IsInDynamicTable()` as Boolean

Description
Test if the whole range is inside a table that supports the different row operations like 'insert', 'append', duplicate, etc.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

IsLastRow

See also

Property: `IsLastRow()` as Boolean (read-only)

Description
Test if the range is in the last row of a table. Which table is taken into consideration depends on the extend of the range. If the selection exceeds a single row of a table, the check is if this table is the last element in an embedding table. See the entry helpers of the user manual for more information.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

IsPasteEnabled

See also

Property: `IsPasteEnabled` as Boolean (read-only)

Description
Checks if the paste operation is supported for this range.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

IsTextStateApplied

See also

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.

LastTextPosition

See also

Property: `LastTextPosition` as Long

Description
Set or get the rightmost text position index of the range object. This index is always greater or equal to `FirstTextPosition`. Indexing starts with 0 at the document beginning, and increments with every different position that the text cursor can occupy. Incrementing the test position by 1, has the same effect as the cursor-right key. Decreasing the test position by 1 has the same effect as the cursor-left key.

If you set LastTextPosition to a value less then the current `FirstTextPosition`, `FirstTextPosition` gets set to the new LastTextPosition .

HINT: Use text cursor positions with care, since this is a costly operation compared to XMLData based cursor positioning.

Errors
2001 The authentic range object, or its related view object is not valid.
2005 Invalid address for the return parameter was specified.
2006 A text position outside the document was specified.

Examples

```
' ---------------------------------------
' XMLSpy scripting environment - VBScript
' ---------------------------------------
Dim objAuthenticView
' we assume that the active document is open in authentic view mode
Set objAuthenticView = Application.ActiveDocument.AuthenticView

nDocStartPosition = objAuthenticView.DocumentBegin.FirstTextPosition
nDocEndPosition = objAuthenticView.DocumentEnd.FirstTextPosition

' let's create a range that selects the whole document
' in an inefficient way
Dim objRange
' we need to get a (any) range object first
Set objRange = objAuthenticView.DocumentBegin
objRange.FirstTextPosition = nDocStartPosition
objRange.LastTextPosition = nDocEndPosition

' let's check if we got it right
If objRange.isEqual(objAuthenticView.WholeDocument) Then
     MsgBox "Test using direct text cursor positioning was ok"
Else
     MsgBox "Oops!"
End If
```

LastXMLData

See also

Property: `LastXMLData` as `XMLData`

© 2007 Altova GmbH

Description
Set or get the last XMLData element in the underlying document that is partially or completely selected by the range. The exact end of the selection is defined by the LastXMLDataOffset attribute.

Whenever you set LastXMLData to a new data object, LastXMLDataOffset gets set to the last cursor position inside this element. Only XMLData objects that have a cursor position may be used. If you set LastXMLData / LastXMLDataOffset, select a position less then the current FirstXMLData / FirstXMLDataOffset, the latter gets moved to the new end position.

HINT: You can use the FirstXMLData and LastXMLData properties to directly access and manipulate the underlying XML document in those cases, where the methods available with the AuthenticRange object are not sufficient.

Errors
2001 The authentic range object, or its related view object is not valid.
2005 Invalid address for the return parameter was specified.
2008 Internal error
2009 The XMLData object cannot be accessed.

LastXMLDataOffset

See also

Property: LastXMLDataOffset as Long

Description
Set or get the cursor position inside LastXMLData element for the end of the range.

Offset positions are based on the characters returned by the Text property and start with 0. When setting a new offset, use -1 to set the offset to the last possible position in the element. The following cases require specific attention:

- The textual form of entries in Combo Boxes, Check Boxes and similar controls can be different from what you see on the screen. Although, the data offset is based on this text, there only two valid offset positions, one at the beginning and one at the end of the entry. An attempt to set the offset to somewhere in the middle of the entry, will result in the offset being set to the end.
- The textual form of XML Entities might differ in length from their representation on the screen. The offset is based on this textual form.

If LastXMLData / LastXMLDataOffset selects a position before FirstXMLData / FirstXMLDataOffset, the latter gets moved to the new end position.

Errors
2001 The authentic range object, or its related view object is not valid.
2005 Invalid offset was specified.
Invalid address for the return parameter was specified.

Examples

```
' ---------------------------------------------
' XMLSpy scripting environment - VBScript
' Select the complete text of an XMLData element
' using XMLData based selection and ExpandTo
' ---------------------------------------------
Dim objAuthenticView
' we assume that the active document is open in authentic view mode
```

```
Set objAuthenticView = Application.ActiveDocument.AuthenticView

' first we use the XMLData based range properties
' to select all text of the first XMLData element
' in the current selection
Dim objRange
Set objRange = objAuthenticView.Selection
objRange.FirstXMLDataOffset = 0  ' start at beginning of element text
objRange.LastXMLData = objRange.FirstXMLData  ' select only one element
objRange.LastXMLDataOffset = -1  ' select till its end

' the same can be achieved with the ExpandTo method
Dim objRange2
Set objRange2 = objAuthenticView.Selection.ExpandTo(spyAuthenticTag)

' were we successful?
If objRange.IsEqual(objRange2) Then
     objRange.Select()
Else
     MsgBox "Ooops"
End If
```

MoveBegin

See also

Method: MoveBegin (*eKind* as SPYAuthenticElementKind, *nCount* as Long) as AuthenticRange

Description

Move the beginning of the range to the beginning of the nCount element of type eKind . Counting starts at the current beginning of the range object.

Use positive numbers for nCount to move towards the document end, use negative numbers to move towards document beginning. The end of the range stays unmoved, unless the new beginning would be larger than it. In this case, the end is moved to the new beginning. The method returns the modified range object.

Errors

2001	The authentic range object, or its related view object is no longer valid.
2003	Target lies after end of document.
2004	Target lies before beginning of document.
2005	Invalid element kind specified. Invalid address for the return parameter was specified.

MoveEnd

See also

Method: MoveEnd (*eKind* as SPYAuthenticElementKind, *nCount* as Long) as AuthenticRange

Description

Move the end of the range to the begin of the nCount element of type eKind . Counting starts at the current end of the range object.

Use positive numbers for nCount to move towards the document end, use negative numbers to move towards document beginning. The beginning of the range stays unmoved, unless the new end would be less than it. In this case, the beginning gets moved to the new end. The method

 © 2007 Altova GmbH

returns the modified range object.

Errors

2001 The authentic range object, or its related view object is no longer valid.
2003 Target lies after end of document.
2004 Target lies before begin of document.
2005 Invalid element kind specified.
Invalid address for the return parameter was specified.

MoveRowDown

See also

Method: `MoveRowDown()` as Boolean

Description
If the beginning of the range is inside a dynamic table and selects a row which is not the last row in this table, this method swaps this row with the row immediately below. The selection of the range moves with the row, but does not otherwise change. The function returns *true* if the move operation was successful, otherwise *false*.

Errors

2001 The authentic range object or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

MoveRowUp

See also

Method: `MoveRowUp()` as Boolean

Description
If the beginning of the range is inside a dynamic table and selects a row which is not the first row in this table, this method swaps this row with the row above. The selection of the range moves with the row, but does not change otherwise. The function returns *true* if the move operation was successful, otherwise *false*.

Errors

2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

Examples

See JScript - Bubble Sort Dynamic Tables.

Parent

See also

Property: `Parent` as `AuthenticView` (read-only)

Description
Access the view that owns this range object.

Errors

2001 The authentic range object, or its related view object is no longer valid.

2005 Invalid address for the return parameter was specified.

Paste

See also

Method: `Paste()` as Boolean

Description
Returns *False* if the copy/paste buffer is empty, or its content cannot replace the current selection.

Otherwise, deletes the current selection, inserts the content of the copy/paste buffer, and returns *True*.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

PerformAction

See also

Method: `PerformAction` (*`eAction`* as `SPYAuthenticActions`, *`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`.

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

```
' --------------------------------------------
' XMLSpy 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
```

 © 2007 Altova GmbH

```
      ' we set a default value and position at its end
      objRange.Text = "Hello"
      objRange.ExpandTo(spyAuthenticTag).CollapsToEnd().Select
Else
      MsgBox "Can't insert any elements at current position"
End If
```

Select

See also

Method: Select()

Description

Makes this range the current user interface selection. You can achieve the same result using: '*objRange.Parent.Selection = objRange*'

Errors

2001 The authentic range object or its related view object is no longer valid.

Examples

```
' ---------------------------------------
' XMLSpy scripting environment - VBScript
' ---------------------------------------
Dim objAuthenticView
' we assume that the active document is open in authentic view mode
Set objAuthenticView = Application.ActiveDocument.AuthenticView

' set current selection to end of document
objAuthenticView.DocumentEnd.Select()
```

SelectNext

See also

Method: SelectNext (*eKind* as SPYAuthenticElementKind) as AuthenticRange

Description

Selects the element of type eKind after the current end of the range. The method returns the modified range object.

Errors

2001 The authentic range object, or its related view object is no longer valid.
2003 Target lies after end of document.
2005 Invalid element kind specified.
Invalid address for the return parameter was specified.

Examples

```
' -------------------------------------------
' XMLSpy scripting environment - VBScript
' Scan through the whole document word-by-word
' -------------------------------------------
Dim objAuthenticView
' we assume that the active document is open in authentic view mode
Set objAuthenticView = Application.ActiveDocument.AuthenticView

Dim objRange
Set objRange = objAuthenticView.DocumentBegin
Dim bEndOfDocument
bEndOfDocument = False
```

```
On Error Resume Next
While Not bEndOfDocument
     objRange.SelectNext(spyAuthenticWord).Select
     If ((Err.number - vbObjecterror) = 2003) Then
            bEndOfDocument = True
            Err.Clear
     ElseIf (Err.number <> 0) Then
            Err.Raise  ' forward error
     End If
Wend
```

SelectPrevious

See also

Method: GotoPrevious (*eKind* as SPYAuthenticElementKind) as AuthenticRange

Description
Selects the element of type eKind before the current beginning of the range. The method returns the modified range object.

Errors

2001	The authentic range object, or its related view object is no longer valid.
2004	Target lies before begin of document.
2005	Invalid element kind specified. Invalid address for the return parameter was specified.

Examples

```
' ------------------------------------------
' XMLSpy scripting environment - VBScript
' Scan through the whole document tag-by-tag
' ------------------------------------------
Dim objAuthenticView
' we assume that the active document is open in authentic view mode
Set objAuthenticView = Application.ActiveDocument.AuthenticView

Dim objRange
Set objRange = objAuthenticView.DocumentEnd
Dim bEndOfDocument
bBeginOfDocument = False

On Error Resume Next
While Not bBeginOfDocument
     objRange.SelectPrevious(spyAuthenticTag).Select
     If ((Err.number - vbObjecterror) = 2004) Then
            bBeginOfDocument = True
            Err.Clear
     ElseIf (Err.number <> 0) Then
            Err.Raise  ' forward error
     End If
Wend
```

SetElementAttributeValue

See also

Method: SetElementAttributeValue (*strElementName* as String, *strAttributeName* as String, *strAttributeValue* as String)

© 2007 Altova GmbH

Description
Retrieve the value of the attribute specified in strAttributeName for the element identified with strElementName. If the attribute is supported but has no value assigned, the empty string is returned. To find out the names of attributes supported by an element, use GetElementAttributeNames, or HasElementAttribute.

Errors
2001 The authentic range object or its related view object is no longer valid.
2005 Invalid element name was specified.
Invalid attribute name was specified.
Invalid attribute value was specified.

Examples

```
' ------------------------------------------
' XMLSpy scripting environment - VBScript
' Get and set element attributes
' ------------------------------------------
Dim objRange
' we assume that the active document is open in authentic view mode
Set objRange = Application.ActiveDocument.AuthenticView.Selection

' first we find out all the elements below the beginning of the range
Dim arrElements
objRange.GetElementHierarchy arrElements

If IsArray(arrElements) Then
     If UBound(arrElements) >= 0 Then
            ' we use the top level element and find out its valid attributes
            Dim arrAttrs()
            objRange.GetElementAttributeNames arrElements(0), arrAttrs

            If UBound(arrAttrs) >= 0 Then
                  ' we retrieve the current value of the first valid
attribute
                  Dim strAttrVal
                  strAttrVal = objRange.GetElementAttributeValue
(arrElements(0), arrAttrs(0))
                  msgbox "current value of " & arrElements(0) & "//" &
arrAttrs(0) & " is: " & strAttrVal

                  ' we change this value and read it again
                  strAttrVal = "Hello"
                  objRange.SetElementAttributeValue arrElements(0),
arrAttrs(0), strAttrVal
                  strAttrVal = objRange.GetElementAttributeValue
(arrElements(0), arrAttrs(0))
                  msgbox "new value of " & arrElements(0) & "//" &
arrAttrs(0) & " is: " & strAttrVal
            End If
     End If
End If
```

SetFromRange

See also

Method: SetFromRange (*objSrcRange* as AuthenticRange)

Description
Sets the range object to the same beginning and end positions as objSrcRange.

Errors
2001 One of the two range objects, is invalid.
2005 Null object was specified as source object.

Text

See also

Property: `Text` as String

Description
Set or get the textual content selected by the range object.

The number of characters retrieved are not necessarily identical, as there are text cursor positions between the beginning and end of the selected range. Most document elements support an end cursor position different to the beginning cursor position of the following element. Drop-down lists maintain only one cursor position, but can select strings of any length. In the case of radio buttons and check boxes, the text property value holds the string of the corresponding XML element.

If the range selects more then one element, the text is the concatenation of the single texts. XML entities are expanded so that '&' is expected as '&'.

Setting the text to the empty string, does not delete any XML elements. Use `Cut`, `Delete` or `PerformAction` instead.

Errors
2001 The authentic range object or its related view object is no longer valid.
2005 Invalid address for a return parameter was specified.

4.3.4 AuthenticView

See also

Properties	Methods	Events
Application	Goto	OnBeforeCopy
AsXMLString	IsRedoEnabled	OnBeforeCut
DocumentBegin	IsUndoEnabled	OnBeforeDelete
DocumentEnd	Print	OnBeforeDrop
Event	Redo	OnBeforePaste
MarkupVisibility	Undo	OnDragOver
Parent	UpdateXMLInstanceEntities	OnKeyBoardEvent
Selection		OnMouseEvent
XMLDataRoot		OnSelectionChanged
WholeDocument		

Description
AuthenticView and its child objects `AuthenticRange` and `AuthenticDataTransfer` provide you with an interface for Authentic, which allow easy and consistent modification of document contents. These interfaces replace the following interfaces which are marked now as **obsolete**:
`OldAuthenticView` (old name was `DocEditView`)
`AuthenticSelection` (old name was `DocEditSelection`, superseded by `AuthenticRange`)
`AuthenticEvent` (old name was `DocEditEvent`)Interfaces

 © 2007 Altova GmbH

AuthenticView gives you easy access to specific features such as printing, the multi-level undo buffer, and the current cursor selection, or position.

AuthenticView uses objects of type AuthenticRange to make navigation inside the document straight-forward, and to allow for the flexible selection of logical text elements. Use the properties DocumentBegin, DocumentEnd, or WholeDocument for simple selections, while using the Goto method for more complex selections. To navigate relative to a given document range, see the methods and properties of the AuthenticRange object.

Examples

```
' ---------------------------------------
' XMLSpy scripting environment - VBScript
' secure access to authentic view object
' ---------------------------------------
Dim objDocument
Set objDocument = Application.ActiveDocument
If (Not objDocument Is Nothing) Then
     ' we have an active document, now check for view mode
     If (objDocument.CurrentViewMode <> spyViewContent) Then
          If (Not objDocument.SwitchViewMode (spyViewContent)) Then
               MsgBox "Active document does not support authentic view
mode"
          Else
               ' now it is safe to access the authentic view object
               Dim objAuthenticView
               Set objAuthenticView = objDocument.AuthenticView
               ' now use the authentic view object

          End If
     End If
Else
     MsgBox "No document is open"
End If
```

Events

OnBeforeCopy

See also

Event: OnBeforeCopy() as Boolean

XMLSpy scripting environment - VBScript:

```
Function On_DocEditCopy()
     'On_BeforeCopy=False 'to disable operation
End Function
```

XMLSpy scripting environment - JScript:

```
function On_DocEditCopy()
{
     //return false;/*to disable operation*/
}
```

XMLSpy IDE Plugin:

```
IXMLSpyPlugIn.OnEvent (21, ...)  //nEventID=21
```

Description

This event gets triggered before a copy operation gets performed on the document. Return *True* (or nothing) to allow copy operation. Return *False* to disable copying.

OnBeforeCut

See also

Event: `OnBeforeCut()` as Boolean

XMLSpy scripting environment - VBScript:

```
Function On_DocEditCut()
	'On_BeforeCopy=False 'to disable operation
End Function
```

XMLSpy scripting environment - JScript:

```
function On_DocEditCut()
{
	//return false;/*to disable operation*/
}
```

XMLSpy IDE Plugin:

```
IXMLSpyPlugIn.OnEvent (20, ...)  //nEventID=20
```

Description

This event gets triggered before a cut operation gets performed on the document. Return *True* (or nothing) to allow cut operation. Return *False* to disable operation.

OnBeforeDelete

See also

Event: `OnBeforeDelete()` as Boolean

XMLSpy scripting environment - VBScript:

```
Function On_DocEditClear()
	'On_BeforeCopy=False 'to disable operation
End Function
```

XMLSpy scripting environment - JScript:

```
function On_DocEditClear()
{
	//return false;/*to disable operation*/
}
```

XMLSpy IDE Plugin:

```
IXMLSpyPlugIn.OnEvent (22, ...)  //nEventID=22
```

Description

This event gets triggered before a delete operation gets performed on the document. Return *True* (or nothing) to allow delete operation. Return *False* to disable operation.

OnBeforeDrop

See also

© 2007 Altova GmbH

Event: `OnBeforeDrop ( i_nXPos as Long, i_nYPos as Long, i_ipRange as AuthenticRange, i_ipData as cancel Boolean`

XMLSpy scripting environment - VBScript:

```
Function On_DocEditDrop(nXPos,nYPos,objRange,objData)
     'On_BeforeCopy=False 'to disable operation
End Function
```

XMLSpy scripting environment - JScript:

```
function On_DocEditDrop(nXPos,nYPos,objRange,objData)
{
     //return false;/*to disable operation*/
}
```

XMLSpy IDE Plugin:

```
IXMLSpyPlugIn.OnEvent (11, ...)  //nEventID=11
```

Description

This event gets triggered whenever a previously dragged object gets dropped inside the application window. All event related information gets passed as parameters.

The first two parameters specify the mouse position at the time when the event occurred. The parameter *objRange* passes a range object that selects the XML element below the mouse position. The value of this parameter might be *NULL*. Be sure to check before you access the range object. The parameter *objData* allows to access information about the object being dragged.

Return *False* to cancle the drop operation. Return *True* (or nothing) to continue normal operation.

Examples

```
'
'------------------------------------------------------------------------------
' VB code snippet - connecting to object level events
'
'------------------------------------------------------------------------------
' access XMLSpy (without checking for any errors)
Dim objSpy As XMLSpyLib.Application
Set objSpy = GetObject("", "XMLSpy.Application")

' this is the event callback routine connected to the OnBeforeDrop
' event of object objView
Private Function objView_OnBeforeDrop(ByVal 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
```

OnBeforePaste

See also

Event: OnBeforePaste (*objData* as Variant, *strType* as String) as Boolean

XMLSpy scripting environment - VBScript:

```
Function On_DocEditPaste(objData, strType)
    'On_BeforeCopy=False 'to disable operation
End Function
```

XMLSpy scripting environment - JScript:

```
function On_DocEditPaste(objData, strType)
{
    //return false; /*to disable operation*/
}
```

XMLSpy 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.

OnDragOver

See also

Event: OnDragOver (*nXPos* as Long, *nYPos* as Long, *eMouseEvent* as SPYMouseEvent, *objRange* as AuthenticRange, *objData* as AuthenticDataTransfer) as Boolean

XMLSpy scripting environment - VBScript:

```
Function On_DocEditDragOver(nXPos, nYPos, eMouseEvent, objRange, objData)
    'On_BeforeCopy=False 'to disable operation
End Function
```

XMLSpy scripting environment - JScript:

```
function On_DocEditDragOver(nXPos, nYPos, eMouseEvent, objRange, objData)
{
    //return false; /*to disable operation*/
}
```

XMLSpy IDE Plugin:

```
IXMLSpyPlugIn.OnEvent (10, ...)   //nEventID=10
```

Description

This event gets triggered whenever an object from within our outside of Authentic gets dragged with the mouse over the application window. All event related information gets passed as parameters.

 © 2007 Altova GmbH

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
```

OnKeyboardEvent

See also

Event: `OnKeyboardEvent` (*`eKeyEvent`* as SPYKeyEvent, *`nKeyCode`* as Long, *`nVirtualKeyStatus`* as Long) as Boolean

XMLSpy scripting environment - VBScript:

```
Function On_DocEditKeyboardEvent(eKeyEvent, nKeyCode, nVirtualKeyStatus
)
    'On_DocEditKeyboardEvent=True 'to cancel bubbling of event
End Function
```

XMLSpy scripting environment - JScript:

```
function On_DocEditKeyboardEvent(eKeyEvent, nKeyCode, nVirtualKeyStatus)
{
```

```
        //return false;/*to cancel bubbling of event*/
}
```

XMLSpy 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, to test for the different keys or their combinations.

REMARK: The following events from the scripting environment and IDE Plugin of XMLSpy are still supported but become obsolete with this event:

```
On_DocEditKeyUp()           IXMLSpyPlugIn.OnEvent (13, ...)    //nEventID=
13
On_DocEditKeyDown()         IXMLSpyPlugIn.OnEvent (12, ...)    //nEventID=
12
On_DocEditKeyPressed()      IXMLSpyPlugIn.OnEvent (14, ...)    //nEventID=
14
```

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 OnKeyboard
' event of object objView
Private Function objView_OnKeyboardEvent(ByVal i_keyEvent As Long, ByVal
io_pnKeyCode As Long, ByVal i_nVirtualKeyStatus As Long) As Boolean
    If ((i_keyEvent = XMLSpyLib.spyKeyUp) And ((i_nVirtualKeyStatus And
XMLSpyLib.spyCtrlKeyMask) <> 0)) Then
        MsgBox ("Ctrl " & io_pnKeyCode & " pressed")
        objView_OnKeyboardEvent = True
    Else
        objView_OnKeyboardEvent = False
    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
```

OnMouseEvent

See also

Event: OnMouseEvent (*nXPos* as Long, *nYPos* as Long, *eMouseEvent* as SPYMouseEvent , *objRange* as AuthenticRange) as Boolean

 © 2007 Altova GmbH

XMLSpy scripting environment - VBScript:

```
Function On_DocEditMouseEvent(nXPos, nYPos, eMouseEvent, objRange)
     'On_DocEditMouseEvent=True 'to cancel bubbling of event
End Function
```

XMLSpy scripting environment - JScript:

```
function On_DocEditMouseEvent(nXPos, nYPos, eMouseEvent, objRange)
{
     //return false; /*to cancel bubbling of event*/
}
```

XMLSpy 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 to test for the different messages, button status, and their combinations.

The parameter *objRange* identifies the part of the document found at the current mouse cursor position. The range objects always selects a complete tag of the document. (This might change in future versions, when a more precise positioning mechanism becomes available). If no selectable part of the document is found at the current position, the range object is *null*.

REMARK: The following events from the scripting environment and IDE Plugin of XMLSpy are still supported but become obsolete with this event:

```
On_DocEditMouseMove()              IXMLSpyPlugIn.OnEvent (15, ...)    //
nEventID=15
On_DocEditButtonUp()               IXMLSpyPlugIn.OnEvent (16, ...)    //
nEventID=16
On_DocEditButtonDown()             IXMLSpyPlugIn.OnEvent (17, ...)    //
nEventID=17
On_DocEditButtonDoubleClick()      IXMLSpyPlugIn.OnEvent (24, ...)    //
nEventID=24
```

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 OnMouseEvent
' event of object objView. If you click with the left mouse button
' while pressing a control key, the current selection will be set
' to the tag below the current mouse cursor position
Private Function objView_OnMouseEvent(ByVal i_nXPos As Long, ByVal i_nYPos
As Long, ByVal i_eMouseEvent As XMLSpyLib.SPYMouseEvent, ByVal i_pRange As
XMLSpyLib.IAuthenticRange) As Boolean
    If (i_eMouseEvent = (XMLSpyLib.spyLeftButtonDownMask Or
XMLSpyLib.spyCtrlKeyDownMask)) Then
        On Error Resume Next
```

```
        i_pRange.Select
        objView_OnMouseEvent = True
    Else
        objView_OnMouseEvent = False
    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
```

OnSelectionChanged

See also

Event: `OnSelectionChanged` (*`objNewSelection`* as `AuthenticRange`)

XMLSpy scripting environment - VBScript:

```
Function On_DocEditSelectionChanged(objNewSelection)
End Function
```

XMLSpy scripting environment - JScript:

```
function On_DocEditSelectionChanged(objNewSelection)
{
}
```

XMLSpy 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
```


© 2007 Altova GmbH

Application

See also

Property: `Application` as `Application` (read-only)

Description
Access the XMLSpy application object.

Errors
2000 The authentic view object is no longer valid.
2005 Invalid address for the return parameter was specified.

AsXMLString

See also

Property: `AsXMLString` as String

Description
Returns or sets the document content as an XML string. Setting the content to a new value does not change the schema file or sps file in use. If the new `XMLString` does not match the actual schema file error 2011 gets returned.

Errors
2000 The authentic view object is no longer valid.
2011 `AsXMLString` was set to a value which is no valid XML for the current schema file.

DocumentBegin

See also

Property: `DocumentBegin` as `AuthenticRange` (read-only)

Description
Retrieve a range object that points to the beginning of the document.

Errors
2000 The authentic view object is no longer valid.
2005 Invalid address for the return parameter was specified.

DocumentEnd

See also

Property: `DocumentEnd` as `AuthenticRange` (read-only)

Description
Retrieve a range object that points to the end of the document.

Errors
2000 The authentic view object is no longer valid.
2005 Invalid address for the return parameter was specified.

Event

See also

Property: `Event` as AuthenticEvent (read-only)

Description

This property gives access to parameters of the last event in the same way as OldAuthenticView.event does. Since all events for the scripting environment and external clients are now available with parameters this `Event` property should only be used from within IDE-Plugins.

Errors

2000 The authentic view object is no longer valid.
2005 Invalid address for the return parameter was specified.

Goto

See also

Method: `Goto` (*`eKind`* as SPYAuthenticElementKind, *`nCount`* as Long, *`eFrom`* as *`SPYAuthenticDocumentPosition`*) as *`AuthenticRange`*

Description

Retrieve a range object that points to the beginning of the *`nCount`* element of type *`eKind`*. The start position is defined by the parameter *`eFrom`*. Use positive values for *`nCount`* to navigate to the document end. Use negative values to navigate towards the beginning of the document.

Errors

2000 The authentic view object is no longer valid.
2003 Target lies after end of document.
2004 Target lies before beginning of document.
2005 Invalid element kind specified.
The document position to start from is not one of *spyAuthenticDocumentBegin* or *spyAuthenticDocumentEnd*.
Invalid address for the return parameter was specified.

Examples

```
' ---------------------------------------
' XMLSpy 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
```


© 2007 Altova GmbH

IsRedoEnabled

See also

Property: `IsRedoEnabled` as Boolean (read-only)

Description
True if redo steps are available and `Redo` is possible.

Errors
2000 The authentic view object is no longer valid.
2005 Invalid address for the return parameter was specified.

IsUndoEnabled

See also

Property: `IsUndoEnabled` as Boolean (read-only)

Description
True if undo steps are available and `Undo` is possible.

Errors
2000 The authentic view object is no longer valid.
2005 Invalid address for the return parameter was specified.

MarkupVisibility

See also

Property: `MarkupVisibility` as `SPYAuthenticMarkupVisibility`

Description
Set or get current visibility of markup.

Errors
2000 The authentic view object is no longer valid.
2005 Invalid enumeration value was specified.
Invalid address for the return parameter was specified.

Parent

See also

Property: `Parent` as `Document` (read-only)

Description
Access the document shown in this view.

Errors
2000 The authentic view object is no longer valid.
2005 Invalid address for the return parameter was specified.

Print

See also

Method: `Print` (*bWithPreview* as Boolean, *bPromptUser* as Boolean)

Description
Print the document shown in this view. If *bWithPreview* is set to *True*, the print preview dialog pops up. If *bPromptUser* is set to *True*, the print dialog pops up. If both parameters are set to *False*, the document gets printed without further user interaction.

Errors
2000 The authentic view object is no longer valid.

Redo

See also

Method: `Redo()` as Boolean

Description
Redo the modification undone by the last undo command.

Errors
2000 The authentic view object is no longer valid.
2005 Invalid address for the return parameter was specified.

Selection

See also

Property: `Selection` as `AuthenticRange`

Description
Set or get current text selection in user interface.

Errors
2000 The authentic view object is no longer valid.
2002 No cursor selection is active.
2005 Invalid address for the return parameter was specified.

Examples

```
' ---------------------------------------
' XMLSpy 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 =
```

 © 2007 Altova GmbH

```
objAuthenticView.Selection.GotoNextCursorPosition()
     ' or shorter:
      objAuthenticView.Selection.GotoNextCursorPosition().Select
End If
```

Undo

See also

Method: Undo() as Boolean

Description
Undo the last modification of the document from within this view.

Errors

2000	The authentic view object is no longer valid.
2005	Invalid address for the return parameter was specified.

UpdateXMLInstanceEntities

See also

Method: UpdateXMLInstanceEntities()

Description
Updates the internal representation of the declared entities, and refills the entry helper. In addition, the validator is reloaded, allowing the XML file to validate correctly. Please note that this may also cause schema files to be reloaded.

Errors
The method never returns an error.

Example

```
// ----------------------------------------
// XMLSpy 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();
     }
}
```

WholeDocument

See also

Property: `WholeDocument` as `AuthenticRange` (read-only)

Description
Retrieve a range object that selects the whole document.

Errors
2000 The authentic view object is no longer valid.
2005 Invalid address for the return parameter was specified.

XMLDataRoot

See also

Property: `XMLDataRoot` as `XMLData` (read-only)

Description
Returns or sets the top-level XMLData element of the current document. This element typically describes the document structure and would be of kind spyXMLDataXMLDocStruct, spyXMLDataXMLEntityDocStruct or spyXMLDataDTDDocStruct..

Errors
2000 The authentic view object is no longer valid.
2005 Invalid address for the return parameter was specified.

4.3.5 CodeGeneratorDlg

See also
Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

Properties and Methods

Standard automation properties
Application
Parent

Programming language selection properties
ProgrammingLanguage
TemplateFileName
CompatibilityMode

Settings for C++ code
CPPSettings_DOMType
CPPSettings_LibraryType
CPPSettings_UseMFC
CPPSettings_GenerateVC6ProjectFile
CPPSettings_GenerateVSProjectFile

Settings for C# code
CSharpSettings_ProjectType

Dialog handling for above code generation properties
PropertySheetDialogAction

Output path selection properties
OutputPath

© 2007 Altova GmbH

OutputPathDialogAction

Presentation of result
OutputResultDialogAction

Description
Use this object to configure the generation of program code for schema files. The method GenerateProgramCode expects a CodeGeneratorDlg as parameter to configure code generation as well as the associated user interactions.

Application

See also
Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

Property: `Application` as `Application` (read-only)

Description
Access the XMLSpy application object.

Errors
2200 The object is no longer valid.
2201 Invalid address for the return parameter was specified.

CompatibilityMode

Property: `CompatibilityMode` as Boolean
Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

Description
Set to true to generate code compatible to XMLSpy 2005R3. Set to false to use newly added code-generation features.

Errors
2200 The object is no longer valid.
2201 Invalid action passed as parameter or an invalid address was specified for the return parameter.

CPPSettings_DOMType

Property: `CPPSettings_DOMType` as `SPYDOMType`
Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

Description
Defines one of the settings that configure generation of C++ code.

Errors
2200 The object is no longer valid.
2201 Invalid action passed as parameter or an invalid address was specified for the return parameter.

CPPSettings_GenerateVC6ProjectFile

Property: `CPPSettings_GenerateVC6ProjectFile` as Boolean
Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

Description
Defines one of the settings that configure generation of C++ code.

Errors
2200 The object is no longer valid.
2201 Invalid action passed as parameter or an invalid address was specified for the return parameter.

CPPSettings_GenerateVSProjectFile

Property: `CSharpSettings_GenerateVSProjectFile` as `SPYProjectType`
Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

Description
Defines one of the settings that configure generation of C++ code. Only `spyVisualStudio2003Project` (=1) and `spyVisualStudio2005Project` (=4) are valid project types.

Errors
2200 The object is no longer valid.
2201 Invalid action passed as parameter or an invalid address was specified for the return parameter.

CPPSettings_LibraryType

Property: `CPPSettings_LibraryType` as `SPYLibType`
Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

Description
Defines one of the settings that configure generation of C++ code.

Errors
2200 The object is no longer valid.
2201 Invalid action passed as parameter or an invalid address was specified for the return parameter.

CPPSettings_UseMFC

Property: `CPPSettings_UseMFC` as Boolean
Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

Description
Defines one of the settings that configure generation of C++ code.

 © 2007 Altova GmbH

Errors

2200 The object is no longer valid.
2201 Invalid action passed as parameter or an invalid address was specified for the return parameter.

CSharpSettings_ProjectType

Property: `CSharpSettings_ProjectType` as `SPYProjectType`
Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

Description
Defines the only setting to configure generation of C# code.

Errors

2200 The object is no longer valid.
2201 Invalid action passed as parameter or an invalid address was specified for the return parameter.

OutputPath

Property: `OutputPath` as String
Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

Description
Selects the base directory for all generated code.

Errors

2200 The object is no longer valid.
2201 Invalid address for the return parameter was specified.

OutputPathDialogAction

Property: `OutputPathDialogAction` as `SPYDialogAction`
Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

Description
Defines how the sub-dialog for selecting the code generation output path gets handled. Set this value to *spyDialogUserInput(2)* to show the dialog with the current value of the OutputPath property as default. Use *spyDialogOK(0)* to hide the dialog from the user.

Errors

2200 The object is no longer valid.
2201 Invalid action passed as parameter or an invalid address was specified for the return parameter.

OutputResultDialogAction

Property: `OutputResultDialogAction` as `SPYDialogAction`
Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

Description
Defines how the sub-dialog that asks to show the result of the code generation process gets handled. Set this value to *spyDialogUserInput(2)* to show the dialog. Use *spyDialogOK(0)* to hide the dialog from the user.

Errors
2200 The object is no longer valid.
2201 Invalid action passed as parameter or an invalid address was specified for the return parameter.

Parent

See also
Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

Property: `Parent` as `Dialogs` (read-only)

Description
Access the parent of the object.

Errors
2200 The object is no longer valid.
2201 Invalid address for the return parameter was specified.

ProgrammingLanguage

Property: `ProgrammingLanguage` as `ProgrammingLanguage`
Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

Description
Selects the output language for the code to be generated.

CAUTION: Setting this property to one of C++, C# or Java, changes the property TemplateFileName to the appropriate template file delivered with XMLSpy as well. If you want to generate C++, C# or Java code based on your own templates, set first the programming language and then select your template file.

Errors
2200 The object is no longer valid.
2201 Invalid address for the return parameter was specified.

PropertySheetDialogAction

Property: `PropertySheetDialogAction` as `SPYDialogAction`
Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

Description
Defines how the sub-dialog that configures the code generation process gets handled. Set this value to *spyDialogUserInput(2)* to show the dialog with the current values as defaults. Use *spyDialogOK(0)* to hide the dialog from the user.

Errors

© 2007 Altova GmbH

2200 The object is no longer valid.
2201 Invalid action passed as parameter or an invalid address was specified for the return parameter.

TemplateFileName

Property: `TemplateFileName` as String
Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

Description
Selects the code generation template file. XMLSpy comes with template files for C++, C# or Java in the SPL folder of your installation directory.

Setting this property to one of the code generation template files of your XMLSpy installation automatically sets the ProgrammingLanguage property to its appropriate value.

Errors
2200 The object is no longer valid.
2201 Invalid address for the return parameter was specified.

4.3.6 DatabaseConnection

See also

Properties for import and export
`File` or
`ADOConnection` or
`ODBCConnection`

Properties for import only
`DatabaseKind`
`SQLSelect`
`AsAttributes`
`ExcludeKeys`
`IncludeEmptyElements`
`NumberDateTimeFormat`
`NullReplacement`
`CommentIncluded`

Properties for export only
`CreateMissingTables`
`CreateNew`
`TextFieldLen`

Description
`DatabaseConnection` specifies the parameters for the database connection.

Please note that the properties of the `DatabaseConnection` interface are referring to the settings of the import and export dialogs of XMLSpy.

ADOConnection

See also

Property: ADOConnection as String

Description
The property ADOConnection contains a connection string. Either use this property or ODBCConnection or File to refer to a database.

Errors
No error codes are returned.

Example

```
Dim objSpyConn As DatabaseConnection
Set objSpyConn = objSpy.GetDatabaseSettings

Dim objADO As DataLinks
Set objADO = CreateObject("DataLinks")

If Not (objADO Is Nothing) Then
 Dim objConn As Connection
 Set objConn = objADO.PromptNew
 objSpyConn.ADOConnection = objConn.ConnectionString
End If
```

AsAttributes

See also

Property: AsAttributes as Boolean

Description
Set AsAttributes to true if you want to initialize all import fields to be imported as attributes. Default is false and will initialize all fields to be imported as elements. This property is used only in calls to Application.GetDatabaseImportElementList.

Errors
No error codes are returned.

CommentIncluded

See also

Property: CommentIncluded as Boolean

Description
This property tells whether additional comments are added to the generated XML. Default is true. This property is used only when importing from databases.

Errors
No error codes are returned.

CreateMissingTables

See also

Property: CreateMissingTables as Boolean

Description

If CreateMissingTables is true, tables which are not already defined in the export database will be created during export. Default is true. This property is used only when exporting to databases.

Errors
No error codes are returned.

CreateNew

See also

Property: CreateNew as Boolean

Description
Set CreateNew true if you want to create a new database on export. Any existing database will be overwritten. See also DatabaseConnection.File. Default is false. This property is used only when exporting to databases.

Errors
No error codes are returned.

DatabaseKind

See also

Property: DatabaseKind as SPYDatabaseKind

Description
Select the kind of database that gets access. The default value is spyDB_Unspecified(7) and is sufficient in most cases. This property is used only when importing from databases.

Errors
No error codes are returned.

ExcludeKeys

See also

Property: ExcludeKeys as Boolean

Description
Set ExcludeKeys to true if you want to exclude all key columns from the import data. Default is false. This property is used only when importing from databases.

Errors
No error codes are returned.

File

See also

Property: File as String

Description

The property File sets the path for the database during export or import. This property can only be used in conjunction with a Microsoft Access database. Either use this property or ODBCConnection or ADOConnection to refer to the database.

See also Import and Export.

Errors
No error codes are returned.

IncludeEmptyElements

See also

Property: IncludeEmptyElements as Boolean

Description
Set IncludeEmptyElements to false if you want to exclude all empty elements. Default is true. This property is used only when importing from databases.

Errors
No error codes are returned.

NullReplacement

See also

Property: NullReplacement as String

Description
This property contains the text value that is used during export for empty elements (null values). Default is "". This property is used only when importing from databases.

Errors
No error codes are returned.

NumberDateTimeFormat

See also

Property: NumberDateTimeFormat as SPYNumberDateTimeFormat

Description
The property NumberDateTimeFormat sets the format of numbers and date- and time-values. Default is spySystemLocale. This property is used only when importing from databases.

Errors
No error codes are returned.

ODBCConnection

See also

Property: ODBCConnection as String

Description

© 2007 Altova GmbH

The property ODBCConnection contains a ODBC connection string. Either use this property or ADOConnection or File to refer to a database.

Errors
No error codes are returned.

SQLSelect

See also

Property: SQLSelect as String

Description
The SQL query for the import is stored in the property SQLSelect. This property is used only when importing from databases. See also Import and Export.

Errors
No error codes are returned.

TextFieldLen

See also

Property: TextFieldLen as long

Description
The property TextFieldLen sets the length for created text fields during the export. Default is 255. This property is used only when exporting to databases.

Errors
No error codes are returned.

4.3.7 Dialogs

See also

Properties and Methods

Standard automation properties
Application
Parent

Various dialog objects
CodeGeneratorDlg
FileSelectionDlg
SchemaDocumentationDlg
GenerateSampleXMLDlg
DTDSchemaGeneratorDlg
FindInFilesDlg
WSDLDocumentationDlg

Description
The Dialogs object provides access to different built-in dialogs of XMLSpy. These dialog objects allow to initialize the fields of user dialogs before they get presented to the user or allow to simulate complete user input by your program.

Application

See also

Property: `Application` as `Application` (read-only)

Description
Access the XMLSpy application object.

Errors
2300 The object is no longer valid.
2301 Invalid address for the return parameter was specified.

CodeGeneratorDlg

See also
Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

Property: `CodeGeneratorDlg` as `CodeGeneratorDlg` (read-only)

Description
Get a new instance of a code generation dialog object. You will need this object to pass the necessary parameters to the code generation methods. Initial values are taken from last usage of the code generation dialog.

Errors
2300 The Dialogs object or one of its parents is no longer valid.
2301 Invalid address for the return parameter was specified.

FileSelectionDlg

See also

Property: `FileSelectionDlg` as `FileSelectionDlg` (read-only)

Description
Get a new instance of a file selection dialog object.

File selection dialog objects are passed to you with the some events that signal opening or saving of documents and projects.

Errors
2300 The Dialogs object or one of its parents is no longer valid.
2301 Invalid address for the return parameter was specified.

Parent

See also

Property: `Parent` as `Application` (read-only)

Description
Access the XMLSpy application object.

© 2007 Altova GmbH

Errors
2300 The object is no longer valid.
2301 Invalid address for the return parameter was specified.

SchemaDocumentationDlg

See also

Property: `SchemaDocumentationDlg` as `SchemaDocumentationDlg` (read-only)

Description
Get a new instance of a dialog object that parameterizes generation of schema documentation. See Document.GenerateSchemaDocumentation for its usage.

Errors
2300 The Dialogs object or one of its parents is no longer valid.
2301 Invalid address for the return parameter was specified.

GenerateSampleXMLDlg

See also

Property: `GenerateSampleXMLDlg` as `GenerateSampleXMLDlg` (read-only)

Description
Get a new instance of a dialog object that parameterizes generation of a sample XML based on a W3C schema or DTD. See GenerateSampleXML for its usage.

Errors
2300 The Dialogs object or one of its parents is no longer valid.
2301 Invalid address for the return parameter was specified.

DTDSchemaGeneratorDlg

See also

Property: `DTDSchemaGeneratorDlg` as `DTDSchemaGeneratorDlg` (read-only)

Description
Get a new instance of a dialog object that parameterizes generation of a schema or DTD. See Document.GenerateDTDOrSchemaEx for its usage.

Errors
2300 The Dialogs object or one of its parents is no longer valid.
2301 Invalid address for the return parameter was specified.

FindInFilesDlg

See also

Property: FindInFilesDlg as FindInFilesDlg (read-only)

Description
Get a new instance of a dialog object that parameterizes the search (or replacement) of strings in files. See Application.FindInFiles for its usage.

Errors
2300 The Dialogs object or one of its parents is no longer valid.
2301 Invalid address for the return parameter was specified.

WSDLDocumentationDlg

See also

Property: `WSDLDocumentationDlg` as `WSDLDocumentationDlg` (read-only)

Description
Get a new instance of a dialog object that parameterizes generation of WSDL documentation. See Document.GenerateWSDLDocumentation for its usage.

Errors
2300 The Dialogs object or one of its parents is no longer valid.
2301 Invalid address for the return parameter was specified.

4.3.8 Document

See also

Properties and Methods

Standard automation properties
Application
Parent

Various document properties and methods
SetActiveDocument
Encoding
SetEncoding (obsolete)
Suggestions

XML validation
IsWellFormed
IsValid
SetExternalIsValid

Document conversion and transformation
AssignDTD
AssignSchema
AssignXSL
AssignXSLFO
ConvertDTDOrSchema
ConvertDTDOrSchemaEx
GenerateDTDOrSchema
GenerateDTDOrSchemaEx
CreateSchemaDiagram
ExecuteXQuery
TransformXSL
TransformXSLFO
GenerateProgramCode (Enterprise Edition only)
GenerateSchemaDocumentation
GenerateSampleXML
GenerateWSDLDocumentation

© 2007 Altova GmbH

Document export
GetExportElementList
ExportToText
ExportToDatabase

File saving and naming
FullName
Name
Path
GetPathName (obsolete)
SetPathName (obsolete)
Title
IsModified
Saved
SaveAs
Save
SaveInString
SaveToURL
Close

View access
CurrentViewMode
SwitchViewMode
AuthenticView
GridView
DocEditView (obsolete)

Access to XMLData
RootElement
DataRoot
CreateChild
UpdateViews
StartChanges
EndChanges
UpdateXMLData

Description
Document objects represent XML documents opened in XMLSpy.

Use one of the following properties to access documents that are already open XMLSpy:
Application.ActiveDocument
Application.Documents

Use one of the following methods to open a new document in XMLSpy:
Documents.OpenFile
Documents.OpenURL
Documents.OpenURLDialog
Documents.NewFile
Documents.NewFileFromText
SpyProjectItem.Open
Application.ImportFromDatabase
Application.ImportFromSchema
Application.ImportFromText
Application.ImportFromWord
Document.ConvertDTDOrSchema
Document.GenerateDTDOrSchema

Events

OnBeforeSaveDocument

See also

Event: OnBeforeSaveDocument(*objDocument* as Document, *objDialog* as FileSelectionDlg)

XMLSpy scripting environment - VBScript:

```
Function On_BeforeSaveDocument(objDocument,objDialog)
End Function

' old handler - now obsolete
' return string to save to new file name
' return empty string to cancel save operation
' return nothing to save to original name
Function On_SaveDocument(objDocument,strFilePath)
End Function
```

XMLSpy scripting environment - JScript:

```
function On_BeforeSaveDocument(objDocument,objDialog)
{
}

// old handler - now obsolete
// return string to save to new file name
// return empty string to cancel save operation
// return nothing to save to original name
function On_SaveDocument(objDocument,strFilePath)
{
}
```

XMLSpy IDE Plugin:

```
IXMLSpyPlugIn.OnEvent (27, ...)  //nEventId=27
```

Description

This event gets fired on any attempt to save a document. The file selection dialog object is initialized with the name chosen for the document file. You can modify this selection. To continue saving the document leave the FileSelectionDlg.DialogAction property of *io_objDialog* at its default value spyDialogOK. To abort saving of the document set this property to spyDialogCancel.

OnBeforeCloseDocument

See also

Event: OnBeforeCloseDocument(*objDocument* as Document) as Boolean

XMLSpy scripting environment - VBScript:

```
Function On_BeforeCloseDocument(objDocument)
     'On_BeforeCloseDocument=False 'to prohibit closing of document
End Function
```

XMLSpy scripting environment - JScript:

```
function On_BeforeCloseDocument(objDocument)
```


© 2007 Altova GmbH

```
{
	//return false;/*to prohibit closing of document*/
}
```

XMLSpy IDE Plugin:

```
IXMLSpyPlugIn.OnEvent (28, ...)  //nEventID=28
```

Description

This event gets fired on any attempt to close a document. To prevent the document from being closed return false.

OnBeforeValidate

See also

Event: `OnBeforeValidate(objDocument as Document, bOnLoading as Boolean, bOnCommand as Boolean) as Boolean`

XMLSpy scripting environment - VBScript:

```
Function On_BeforeValidate(objDocument, bOnLoading, bOnCommand)
	On_BeforeValidate=bCancelDefaultValidation		'set by the script if
necessary
End Function
```

XMLSpy scripting environment - JScript:

```
function On_BeforeValidate(objDocument, eViewMode, bActivated)
{
	return bCancelDefaultValidation		//set by the script if necessary
}
```

XMLSpy IDE Plugin:

```
IXMLSpyPlugIn.OnEvent (32, ...)  //nEventID=32
```

Description

This event gets fired before the document is validated. It is possible to suppress the default validation by returning false from the event handler. In this case the script should also set the validation result using the SetExternalIsValid method.

`bOnLoading` is true if the the event is raised on the initial validation on loading the document.

`bOnCommand` is true whenever the user selected the Validate command from the Toolbar or menu.

Available with TypeLibrary version 1.5

OnCloseDocument

See also

Event: `OnCloseDocument(objDocument as Document)`

XMLSpy scripting environment - VBScript:

```
Function On_Close Document(objDocument)
End Function
```

XMLSpy scripting environment - JScript:

```
function On_Close Document(objDocument)
{
}
```

XMLSpy IDE Plugin:

```
IXMLSpyPlugIn.OnEvent (8, ...)    //nEventID=8
```

Description

This event gets fired as a result of closing a document. Do not modify the document from within this event.

OnViewActivation

See also

Event: `OnViewActivation(objDocument` as Document, *eViewMode* as SPYViewModes, *bActivated* as Boolean)

XMLSpy scripting environment - VBScript:

```
Function On_ViewActivation(objDocument,eViewMode,bActivated)
End Function
```

XMLSpy scripting environment - JScript:

```
function On_ViewActivation(objDocument,eViewMode,bActivated)
{
}
```

XMLSpy IDE Plugin:

```
IXMLSpyPlugIn.OnEvent (29, ...)    //nEventID=29
```

Description

This event gets fired whenever a view of a document becomes visible (i.e. becomes the active view) or invisible (i.e. another view becomes the active view or the document gets closed). However, the first view activation event after a document gets opened cannot be received, since there is no document object to get the event from. Use the Application.OnDocumentOpened event instead.

Application

See also

Property: `Application` as Application (read-only)

Description

Access the XMLSpy application object.

Errors

1400 The object is no longer valid.
1407 Invalid address for the return parameter was specified.

© 2007 Altova GmbH

AssignDTD

See also

Method: AssignDTD(*strDTDFile* as String, *bDialog* as Boolean)

Description
The method places a reference to the DTD file "strDTDFile" into the document. Note that no error occurs if the file does not exist, or is not accessible. If bDialog is true XMLSpy presents a dialog to set the file.

See also Simple document access.

Errors
1400 The object is no longer valid.
1409 You are not allowed to assign a DTD to the document.

AssignSchema

See also

Method: AssignSchema (*strSchemaFile* as String, *bDialog* as Boolean)

Description
The method places a reference to the schema file "strSchemaFile" into the document. Note that no error occurs if the file does not exist or is not accessible. If bDialog is true XMLSpy presents a dialog to set the file.

See also Simple document access.

Errors
1400 The object is no longer valid.
1409 You are not allowed to assign a schema file to the document.

AssignXSL

See also

Method: AssignXSL (*strXSLFile* as String, *bDialog* as Boolean)

Description
The method places a reference to the XSL file "strXSLFile" into the document. Note that no error occurs if the file does not exist or is not accessible. If bDialog is true XMLSpy presents a dialog to set the file.

Errors
1400 The object is no longer valid.
1409 You are not allowed to assign an XSL file to the document.

AssignXSLFO

See also

Method: AssignXSLFO (*strXSLFOFile* as String, *bDialog* as Boolean)

Description
The method places a reference to the XSLFO file "strXSLFile" into the document. Note that no error occurs if the file does not exist or is not accessible. If `bDialog` is true XMLSpy presents a dialog to set the file.

Errors
1400 The object is no longer valid.
1409 You are not allowed to assign an XSL file to the document.

AuthenticView

See also

Method: `AuthenticView` as `AuthenticView` (read-only)

Description
Returns an object that gives access to properties and methods specific to Authentic view. The object returned is only valid if the current document is opened in Authentic view mode. The lifetime of an object ends with the next view switch. Any attempt to access objects or any of its children afterwards will result in an error indicating that the object is invalid.

`AuthenticView` and `DocEditView` both provide automation access to the Authentic view mode of XMLSpy. Functional overlap is intentional. A future version of Authentic View will include all functionality of `DocEditView` and its sub-objects, thereby making usage of `DocEditView` obsolete.

Errors
1400 The object is no longer valid.
1417 Document needs to be open in authentic view mode.

Examples

```
' ----------------------------------------
' XMLSpy scripting environment - VBScript
' secure access to authentic view object
' ----------------------------------------
Dim objDocument
Set objDocument = Application.ActiveDocument
If (Not objDocument Is Nothing) Then
	' we have an active document, now check for view mode
	If (objDocument.CurrentViewMode <> spyViewContent) Then
		If (Not objDocument.SwitchViewMode (spyViewContent)) Then
			MsgBox "Active document does not support authentic view
mode"
		Else
			' now it is safe to access the authentic view object
			Dim objAuthenticView
			Set objAuthenticView = objDocument.AuthenticView
			' now use the authentic view object

		End If
	End If
Else
	MsgBox "No document is open"
End If
```

Close

See also

© 2007 Altova GmbH

Method: `Close` (*`bDiscardChanges`* as Boolean)

Description
To close the document call this method. If `bDiscardChanges` is true and the document is modified, the document will be closed but not saved.

Errors
1400 The object is no longer valid.
1401 Document needs to be saved first.

ConvertDTDOrSchema

See also

Method: `ConvertDTDOrSchema` (*`nFormat`* as `SPYDTDSchemaFormat`, *`nFrequentElements`* as `SPYFrequentElements`)

Parameters
`nFormat`
Sets the schema output format to DTD, or W3C.

`nFrequentElements`
Create complex elements as elements or complex types.

Description
`ConvertDTDOrSchema` takes an existing schema format and converts it into a different format. For a finer tuning of DTD / schema conversion, use ConvertDTDOrSchemaEx.

Errors
1400 The object is no longer valid.
1412 Error during conversion.

ConvertDTDOrSchemaEx

See also

Method: `ConvertDTDOrSchemaEx` (*`nFormat`* as `SPYDTDSchemaFormat`, *`nFrequentElements`* as `SPYFrequentElements`, `sOutputPath` as `String`, `nOutputPathDialogAction` as `SPYDialogAction`)

Parameters
`nFormat`
Sets the schema output format to DTD, or W3C.

`nFrequentElements`
Create complex elements as elements or complex types.

sOutputPath
The file path for the newly generated file.

nOutputPathDialogAction
Defines the dialog interaction for this call.

Description
`ConvertDTDOrSchemaEx` takes an existing schema format and converts it into a different format.

Errors
1400 The object is no longer valid.
1412 Error during conversion.

CreateChild

See also

Method: `CreateChild` (*`nKind`* as `SPYXMLDataKind`) as `XMLData`

Return Value
The method returns the new `XMLData` object.

Description
To create a new `XMLData` object use the `CreateChild()` method. See also Using XMLData.

Errors
1400 The object is no longer valid.
1404 Cannot create XMLData object.
1407 Invalid address for the return parameter was specified.

CreateSchemaDiagram

See also

Method: `CreateSchemaDiagram` (*`nKind`* as `SPYSchemaDefKind`, *`strName`* as String, *`strFile`* as String)

Return Value
None.

Description
The method creates a diagram of the schema type `strName` of kind `nKind` and saves the output file into `strFile`.

Errors
1400 The object is no longer valid.
1414 Failed to save diagram.
1415 Invalid schema definition type specified.

CurrentViewMode

See also

Method: `CurrentViewMode` as `SPYViewModes`

Description
The property holds the current view mode of the document. See also `Document.SwitchViewMode`.

Errors
1400 The object is no longer valid.
1407 Invalid address for the return parameter was specified.

© 2007 Altova GmbH

DataRoot

See also

Property: `DataRoot` as `XMLData` (read-only)

Description
This property provides access to the document's first XMLData object of type *spyXMLDataElement*. This is typically the root element for all document content data. See XMLSpyDocument.RootElement to get the root element of the whole document including XML prolog data.

Errors
1400 The document object is no longer valid.
1407 Invalid address for the return parameter was specified.

DocEditView

See also

Method: `DocEditView` as `DocEditView`

Description
Holds a reference to the current Authentic object.

Errors
1400 The object is no longer valid.
1407 Invalid address for the return parameter was specified.
1417 Document needs to be open in authentic view mode.

Encoding

See also

Property: `Encoding` as String

Description
This property provides access to the document's encoding value. However, this property can only be accessed when the document is opened in *spyViewGrid*, *spyViewText* or *spyViewContent*. See CurrentViewMode on how to detect that a document's actual view mode.

This property makes the method SetEncoding obsolete.

Possible values are, for example:

8859-1,
8859-2,
ASCII, ISO-646,
850,
1252,
1255,
SHIFT-JIS, MS-KANJI,
BIG5, FIVE,
UTF-7,
UTF-8,

UTF-16

Errors
1400 The document object is no longer valid.
1407 Invalid address for the return parameter was specified.
1416 Operation not supported in current view mode.

EndChanges

See also

Method: `EndChanges()`

Description
Use the method `EndChanges` to display all changes since the call to `Document.StartChanges`.

Errors
1400 The object is no longer valid.

ExecuteXQuery

See also

Method: `ExecuteXQuery` (*`strXMLFileName`* as String)

Description
Execute the XQuery statements contained in the document. Use the XML file specified as XML source for the transformation. If your XQuery script does not use an XML source set the parameter `strXMLFileName` to an empty string.

Errors
1400 The document object is no longer valid.
1423 XQuery transformation error.
1424 Not all files required for operation could be loaded. Most likely, the file specified in `strXMLFileName` does not exist or is not valid.

ExportToDatabase

See also

Method: `ExportToDatabase` (*`pFromChild`* as `XMLData`, *`pExportSettings`* as `ExportSettings`, *`pDatabase`* as `DatabaseConnection`)

Description
`ExportToDatabase` exports the XML document starting with the element `pFromChild`. The parameter `pExportSettings` defines the behaviour of the export (see `Application.GetExportSettings`). The parameter `pDatabase` specifies the destination of the export (see `Application.GetDatabaseSettings`).

Errors
1400 The object is no longer valid.
1407 Invalid parameter or invalid address for the return parameter was specified.

© 2007 Altova GmbH

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
```

ExportToText

See also

Method: ExportToText (*pFromChild* as XMLData, *pExportSettings* as ExportSettings, *pTextSettings* as TextImportExportSettings)

Description

ExportToText exports tabular information from the document starting at pFromChild into one or many text files. Columns of the resulting tables are generated in alphabetical order of the column header names. Use GetExportElementList to learn about the data that will be exported. The parameter pExportSettings defines the specifics for the export. Set the property ExportSettings.ElementList to the - possibly modified - list returned by GetExportElementList to avoid exporting all contained tables. The parameter pTextSettings defines the options specific to text export and import. You need to set the property TextImportExportSettings.DestinationFolder before you call ExportToText.

See also Import and export of data.

Errors

1400 The object is no longer valid.
1407 Invalid parameter or invalid address for the return parameter was specified.
1416 Error during export.
1430 Document export failed.

Example

```
' ----------------------------------------------------------
' VBA client code fragment - export document to text files
' ----------------------------------------------------------
Dim objDoc As Document
Set objDoc = objSpy.ActiveDocument
```

```
Dim objExpSettings As ExportSettings
Set objExpSettings = objSpy.GetExportSettings
objExpSettings.ElementList = objDoc.GetExportElementList(
        objDoc.RootElement,
        objExpSettings)

Dim objTextExp As TextImportExportSettings
Set objTextExp = objSpy.GetTextExportSettings
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
```

FullName

See also

Property: `FullName` as String

Description
This property can be used to get or set the full file name - including the path - to where the document gets saved. The validity of the name is not verified before the next save operation.

This property makes the methods GetPathName and SetPathName obsolete.

Errors
1400 The document object is no longer valid.
1402 Empty string has been specified as full file name.

GenerateDTDOrSchema

See also

Method: `GenerateDTDOrSchema` (*`nFormat`* as `SPYDTDSchemaFormat`, *`nValuesList`* as integer, *`nDetection`* as `SPYTypeDetection`, *`nFrequentElements`* as `SPYFrequentElements`)

Parameters
`nFormat`
Sets the schema output format to DTD, or W3C.

`nValuesList`
Generate not more than this amount of enumeration-facets per type. Set to -1 for unlimited.

`nDetection`
Specifies granularity of simple type detection.

`nFrequentElements`
Shall the types for all elements be defined as global? Use that value *spyGlobalComplexType* to define them on global scope. Otherwise, use the value *spyGlobalElements*.

Description

 © 2007 Altova GmbH

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.

Errors

1400 The object is no longer valid.
1407 Invalid parameter or invalid address for the return parameter was specified.

GenerateDTDOrSchemaEx

See also

Method: `GenerateDTDOrSchemaEx` (*objDlg* as DTDSchemaGeneratorDlg) as `Document`

Description

Use this method to automatically generate a DTD or schema for the current XML document. A DTDSchemaGeneratorDlg object is used to pass information to the schema/DTD generator. The generation process can be configured to allow user interaction or run without further user input.

Errors

1400 The object is no longer valid.
1407 Invalid parameter or invalid address for the return parameter was specified.

GenerateProgramCode

Method: `GenerateProgramCode` (*objDlg* as `CodeGeneratorDlg`)
Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

Description

Generate Java, C++ or C# class files from the XML Schema definitions in your document. A `CodeGeneratorDlg` object is used to pass information to the code generator. The generation process can be configured to allow user interaction or run without further user input.

Errors

1400 The document object is no longer valid.
1407 An empty file name has been specified.
1421 Feature not available in this edition

GenerateSampleXML

Method: `GenerateSampleXML` (*objDlg* as `GenerateSampleXMLDlg`) as `Document`

Description

Generates a sample XML if the document is a schema or DTD. Use Dialogs.GenerateSampleXMLDlg to get an initialized set of options.

Available with TypeLibrary version 1.5

Errors

1400 The document object is no longer valid.

GenerateSchemaDocumentation

Method: GenerateSchemaDocumentation (*objDlg* as SchemaDocumentationDlg)

Description
Generate documentation for a schema definition file in HTML, MS-Word, or RTF format. The parameter objDlg is used to parameterize the generation process. Use Dialogs.SchemaDocumentationDlg to get an initialized set of options. As a minimum, you will need to set the property SchemaDocumentationDlg.OutputFile before starting the generation process.

Errors

1400	The document object is no longer valid.
1407	Invalid parameters have been passed or an empty file name has been specified as output target.
1417	The document is not opened in schema view, maybe it is not an '.xsd' file.
1421	Feature is not available in this edition.
1422	Error during generation

GenerateWSDLDocumentation

Method: GenerateWSDLDocumentation (*objDlg* as WSDLDocumentationDlg)

Description
Generate documentation for a WSDL definition file in HTML, MS-Word, or RTF format. The parameter objDlg is used to parameterize the generation process. Use Dialogs.WSDLDocumentationDlg to get an initialized set of options. As a minimum, you will need to set the property WSDLDocumentationDlg.OutputFile before starting the generation process.

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

GetExportElementList

See also

Method: GetExportElementList (*pFromChild* as XMLData, *pExportSettings* as ExportSettings) as ElementList

Description
GetExportElementList creates a collection of elements to export from the document, depending on the settings in pExportSettings and starting from the element pFromChild. The function returns a collection of ElementListItems where the properties ElementListItem.Name contain the names of the tables that can be exported from the document. The property ElementListItem.FieldCount contains the number of columns in the table. The property ElementListItem.RecordCount contains the number of records in

 © 2007 Altova GmbH

the table. The property ElementListItem.ElementKind is unused.

See also Import and export of data.

Errors

1400 The object is no longer valid.
1407 Invalid parameter or invalid address for the return parameter was specified.
1427 Failed creating parser for the specified XML.
1428 Export of element list failed.

GetPathName (obsolete)

Superseded by Document.FullName

```
// ----- javascript sample -----
// instead of:
// strPathName = Application.ActiveDocument.GetPathName();
// use now:
strPathName = Application.ActiveDocument.FullName;
```

See also

Method: GetPathName() as String

Description
The method GetPathName gets the path of the active document.

See also Document.SetPathName (obsolete).

GridView

See also

Property: GridView as GridView

Description
This property provides access to the grid view functionality of the document.

Errors

1400 The object is no longer valid.
1407 Invalid address for the return parameter was specified.
1417 Document needs to be open in enhanced grid view mode.

IsModified

See also

Property: IsModified as Boolean

Description
True if the document is modified.

Errors

1400 The object is no longer valid.
1407 Invalid address for the return parameter was specified.

IsValid

See also

Method: `IsValid` (*`strError`* as Variant, *`nErrorPos`* as Variant, *`pBadData`* as Variant) as Boolean

Return Value

True if the document is valid, false if not.

Description

`IsValid()` validates the document against its associated schema or DTD. `strError` and `nErrorPos` give you the same information as XMLSpy if you validate the file within the editor.

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.

Examples

For an example written in VBA see Overview - Simple document access

```
' ----------------------------------------
' XMLSpy scripting environment - VBScript
' ----------------------------------------
Dim errorText
Dim errorPos
Dim badData

' validate the active document
Set doc = Application.ActiveDocument
valid = doc.IsValid(errorText, errorPos, badData)

If (Not valid) Then
       ' create the validation error message text
       Text = errorText

       If ( Not IsEmpty(badData) ) Then
              Text = Text & "(" & badData.Name & "/" & badData.TextValue & ")"
       End If

       Call MsgBox("validation error[" & errorPos & "]: " & Text)
End If

// ----------------------------------------
// XMLSpy scripting environment - JScript
// ----------------------------------------
// 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);

// validate the active document
```


© 2007 Altova GmbH

```
var doc = Application.ActiveDocument;
var valid = doc.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 + ")"
;

	MsgBox("validation error[" + errorPos + "]: " + text);
}
```

IsWellFormed

See also

Method: `IsWellFormed` (*`pData`* as `XMLData`, *`bWithChildren`* as `Boolean`, *`strError`* as Variant, *`nErrorPos`* as Variant, *`pBadXMLData`* as Variant) as Boolean

Return Value

True if the document is well formed.

Description

`IsWellFormed` checks the document for well-formedness starting at the element `pData`.

If the document is not well formed, `strError` contains an error message, `nErrorPos` the position in the file and `pBadXMLData` holds a reference to the element which breaks the well-formedness. These out-parameters are defined as VARIANTs to support scripting languages like VBScript.

Errors

1400 The object is no longer valid.
1407 Invalid parameter or invalid address for the return parameter was specified.

Example

See IsValid.

Name

See also

Property: `Name` as String (read-only)

Description

Use this property to retrieve the name - not including the path - of the document file. To change the file name for a document use the property FullName.

Errors

1400 The document object is no longer valid.
1407 Invalid address for the return parameter was specified.

Parent

See also

Property: `Parent` as `Documents` (read-only)

Description
Access the parent of the document object.

Errors
2400 The document object is no longer valid.
2401 Invalid address for the return parameter was specified.

Path

See also

Property: `Path` as String (read-only)

Description
Use this property to retrieve the path - not including the file name - of the document file. To change the file name and path for a document use the property FullName.

Errors
1400 The document object is no longer valid.
1407 Invalid address for the return parameter was specified.

RootElement

See also

Property: `RootElement` as `XMLData` (read-only)

Description
The property `RootElement` provides access to the root element of the XML structure of the document including the XML prolog data. To access the first element of a document's content navigate to the first child of kind *spyXMLDataElement* or use the Document.DataRoot property.

Errors
1400 The document object is no longer valid.
1407 Invalid address for the return parameter was specified.

Save

See also

Method: `Save()`

Description
The method writes any modifications of the document to the associated file. See also `Document.FullName`.

Errors
1400 The document object is no longer valid.
1407 An empty file name has been specified.

© 2007 Altova GmbH

1403 Error when saving file, probably the file name is invalid.

SaveAs

See also

Method: `SaveAs` (*`strFileName`* as String)

Description
Save the document to the file specified. If saving was successful, the FullName property gets set to the specified file name.

Errors
1400 The document object is no longer valid.
1407 An empty file name has been specified.
1403 Error when saving file, probably the file name is invalid.

Saved

See also

Property: `Saved` as Boolean (read-only)

Description
This property can be used to check if the document has been saved after the last modifications. It returns the negation of IsModified.

Errors
1400 The document object is no longer valid.
1407 Invalid address for the return parameter was specified.

SaveInString

See also

Method: `SaveInString` (*`pData`* as `XMLData`, *`bMarked`* as Boolean) as String

Parameters
`pData`
`XMLData` element to start. Set `pData` to `Document.RootElement` 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.

Errors
1400 The object is no longer valid.

1407 Invalid parameter or invalid address for the return parameter was specified.

SaveToURL

See also

Method: SaveToURL (*strURL* as String, *strUser* as String, *strPassword* as String)

Return Value

Description
SaveToURL() writes the document to the URL strURL. This method does not set the permanent file path of the document.

Errors
1400 The object is no longer valid.
1402 Invalid URL specified.
1403 Error while saving to URL.

SetActiveDocument

See also

Method: SetActiveDocument()

Description
The method sets the document as the active and brings it to the front.

Errors
1400 The object is no longer valid.

SetEncoding (obsolete)

Superseded by Document.Encoding

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.SetEncoding("UTF-16");
// use now:
Application.ActiveDocument.Encoding = "UTF-16";
```

See also

Method: SetEncoding (*strEncoding* as String)

Description
SetEncoding sets the encoding of the document like the menu item "File/Encoding..." in XMLSpy. Possible values for strEncoding are, for example:

8859-1,
8859-2,
ASCII, ISO-646,
850,
1252,

 © 2007 Altova GmbH

1255,
SHIFT-JIS, MS-KANJI,
BIG5, FIVE,
UTF-7,
UTF-8,
UTF-16

SetExternalIsValid

See also

Method: `SetExternalIsValid` (*`bValid`* as Boolean)

Parameters

`bValid`
Sets the result of an external validation process.

Description
The internal information set by this method is only queried on cancelling the default validation in any OnBeforeValidate handler.

Available with TypeLibrary version 1.5

Errors
1400 The object is no longer valid.

SetPathName (obsolete)

Superseded by Document.FullName

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.SetPathName("C:\\myXMLFiles\\test.xml");
// use now:
Application.ActiveDocument.FullName = "C:\\myXMLFiles\\test.xml";
```

See also

Method: `SetPathName` (*`strPath`* as String)

Description
The method `SetPathName` sets the path of the active document. `SetPathName` only copies the string and does not check if the path is valid. All succeeding save operations are done into this file.

StartChanges

See also

Method: `StartChanges()`

Description
`After StartChanges` is executed XMLSpy will not update its editor windows until `Document.EndChanges` is called. This increases performance of complex tasks to the XML structure.

Errors
1400 The object is no longer valid.

© 2007 Altova GmbH

Suggestions

Property: Suggestions as Array

Description
This property contains the last valid user suggestions for this document. The XMLSpy generated suggestions can be modified before they are shown to the user in the OnBeforeShowSuggestions event.

Errors
1400 The object is no longer valid.
1407 Invalid parameter or invalid address for the return parameter was specified.

SwitchViewMode

See also

Method: `SwitchViewMode` (*nMode* as `SPYViewModes`) as Boolean

Return value
Returns true if view mode is switched.

Description
The method sets the current view mode of the document in XMLSpy. See also `Document.CurrentViewMode`.

Errors
1400 The object is no longer valid.
1407 Invalid address for the return parameter was specified.
1417 Invalid view mode specified.

TextView

See also

Property: TextView as TextView

Description
This property provides access to the text view functionality of the document.

Errors
1400 The object is no longer valid.
1407 Invalid address for the return parameter was specified.

Title

See also

Property: `Title` as String (read-only)

Description
`Title` contains the file name of the document. To get the path and filename of the file use `FullName`.

Errors
1400 The document object is no longer valid.
1407 Invalid address for the return parameter was specified.

TransformXSL

See also

Method: `TransformXSL()`

Description
`TransformXSL` processes the XML document via the associated XSL file. See `Document.AssignXSL` on how to place a reference to a XSL file into the document.

Errors
1400 The document object is no longer valid.
1411 Error during transformation process.

TransformXSLFO

See also

Method: `TransformXSLFO()`

Description
`TransformXSLFO` processes the XML document via the associated XSLFO file. See `AssignXSLFO` on how to place a reference to a XSLFO file into the document. You need to assign a FOP processor to XMLSpy before you can use this method.

Errors
1400 The document object is no longer valid.
1411 Error during transformation process.

UpdateViews

See also

Method: `UpdateViews()`

Description
To redraw the Enhanced Grid View and the Tree View call `UpdateViews`. This can be important after you changed the `XMLData` structure of a document. This method does not redraw the text view of XMLSpy.

Errors
1400 The document object is no longer valid.

UpdateXMLData

See also

Method: `UpdateXMLData()` as Boolean

© 2007 Altova GmbH

Description
The XMLData tree is updated from the current view. Please note that this can fail in case of the TextView if the current XML text is not well-formed.

Available with TypeLibrary version 1.5

Errors
1400 The document object is no longer valid.

4.3.9 Documents

See also

Properties
Count
Item

Methods
NewFile
OpenFile
OpenURL
OpenURLDialog
NewFileFromText

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);
}
```

Count

See also

Property: `Count` as long

Description
Count of open documents.

Item

See also

Method: `Item` (*n* as long) as `Document`

Description
Gets the document with the index n in this collection. Index is 1-based.

NewFile

See also

Method: `NewFile` (*strFile* as String, *strType* as String) as `Document`

Parameters
strFile
Full path of new file.

strType
Type of new file as string (i.e. "xml", "xsd", ...)

Return Value
Returns the new file.

Description
`NewFile` creates a new file of type strType (i.e. "xml"). The newly created file is also the ActiveDocument.

NewFileFromText

See also

Method: `NewFileFromText` (*strText* as String, *strType* as String) as `Document`

Parameters
strText
The content of the new document in plain text.

strType
Type of the document to create (i.e. "xml").

Return Value
The method returns the new document.

Description
NewFileFromText creates a new document with strText as its content.

© 2007 Altova GmbH

OpenFile

See also

Method: OpenFile (*strPath* as String, *bDialog* as Boolean) as Document

Parameters

strPath
Path and file name of file to open.

bDialog
Show dialogs for user input.

Return Value

Returns the opened file on success.

Description

OpenFile opens the file strPath. If bDialog is TRUE, a file-dialog will be displayed.

Example

```
Dim objDoc As Document
Set objDoc = objSpy.Documents.OpenFile(strFile, False)
```

OpenURL

See also

Method: OpenURL (*strURL* as String, *nURLType* as , *nLoaSPYURLTypesing* as SPYLoading, *strUser* as String, *strPassword* as String) as Document

Parameters

strURL
URL to open as document.

nURLType
Type of document to open. Set to -1 for auto detection.

nLoading
Set nLoading to 0 (zero) if you want to load it from cache or proxy. Otherwise set nLoading to 1.

strUser
Name of the user if required. Can be empty.

strPassword
Password for authentification. Can be empty.

Return Value

The method returns the opened document.

Description

OpenURLopens the URL strURL.

OpenURLDialog

See also

Method: OpenURLDialog (*strURL* as String, *nURLType* as SPYURLTypes, *nLoading* as SPYLoading, *strUser* as String, *strPassword* as String) as Document

Parameters

strURL
URL to open as document.

nURLType
Type of document to open. Set to -1 for auto detection.

nLoading
Set nLoading to 0 (zero) if you want to load it from cache or proxy. Otherwise set nLoading to 1.

strUser
Name of the user if required. Can be empty.

strPassword
Password for authentification. Can be empty.

Return Value
The method returns the opened document.

Description
OpenURLDialog displays the "open URL" dialog to the user and presets the input fields with the given parameters.

4.3.10 DTDSchemaGeneratorDlg

See also

Properties and Methods

Standard automation properties
Application
Parent

DTDSchemaFormat
ValueList
TypeDetection
FrequentElements
MergeAllEqualNamed
ResolveEntities
AttributeTypeDefinition
GlobalAttributes
OnlyStringEnums
MaxEnumLength
OutputPath
OutputPathDialogAction

Description
Use this object to configure the generation of a schema or DTD. The method GenerateDTDOrSchemaEx expects a DTDSchemaGeneratorDlg as parameter to configure the

 © 2007 Altova GmbH

generation as well as the associated user interactions.

Application

Property: `Application` as `Application` (read-only)

Description
Access the XMLSpy application object.

Errors
3000 The object is no longer valid.
3001 Invalid address for the return parameter was specified.

AttributeTypeDefinition

Property: AttributeTypeDefinition as `SPYAttributeTypeDefinition`

Description
Specifies how attribute definitions get merged.

Errors
3000 The object is no longer valid.
3001 Invalid address for the return parameter was specified.

DTDSchemaFormat

Property: DTDSchemaFormat as `SPYDTDSchemaFormat`

Description
Sets the schema output format to DTD, or W3C.

Errors
3000 The object is no longer valid.
3001 Invalid address for the return parameter was specified.

FrequentElements

Property: FrequentElements as `SPYFrequentElements`

Description
Shall the types for all elements be defined as global? Use that value *spyGlobalComplexType* to define them on global scope. Otherwise, use the value *spyGlobalElements*.

Errors
3000 The object is no longer valid.
3001 Invalid address for the return parameter was specified.

GlobalAttributes

Property: GlobalAttributes as Boolean

Description

Shall attributes with same name and type be resolved globally?

Errors
3000 The object is no longer valid.
3001 Invalid address for the return parameter was specified.

MaxEnumLength

Property: MaxEnumLength as Integer

Description
Specifies the maximum number of characters allowed for enumeration names. If one value is longer than this, no enumeration will be generated.

Errors
3000 The object is no longer valid.
3001 Invalid address for the return parameter was specified.

MergeAllEqualNamed

Property: MergeAllEqualNamed as Boolean

Description
Shall types of all elements with the same name be merged into one type?

Errors
3000 The object is no longer valid.
3001 Invalid address for the return parameter was specified.

OnlyStringEnums

Property: OnlyStringEnums as Boolean

Description
Specifies if enumerations will be created only for plain strings or all types of values.

Errors
3000 The object is no longer valid.
3001 Invalid address for the return parameter was specified.

OutputPath

Property: OutputPath as String

Description
Selects the file name for the generated schema/DTD.

Errors
3000 The object is no longer valid.
3001 Invalid address for the return parameter was specified.

© 2007 Altova GmbH

OutputPathDialogAction

Property: OutputPathDialogAction as `SPYDialogAction`

Description
Defines how the sub-dialog for selecting the schema/DTD output path gets handled. Set this value to *spyDialogUserInput(2)* to show the dialog with the current value of the OutputPath property as default. Use *spyDialogOK(0)* to hide the dialog from the user.

Errors
3000 The object is no longer valid.
3001 Invalid address for the return parameter was specified.

Parent

Property: `Parent` as `Dialogs` (read-only)

Description
Access the parent of the object.

Errors
3000 The object is no longer valid.
3001 Invalid address for the return parameter was specified.

ResolveEntities

Property: ResolveEntities as Boolean

Description
Shall all entities be resolved before generation starts? If yes, an info-set will be built.

Errors
3000 The object is no longer valid.
3001 Invalid address for the return parameter was specified.

TypeDetection

Property: TypeDetection as `SPYTypeDetection`

Description
Specifies granularity of simple type detection.

Errors
3000 The object is no longer valid.
3001 Invalid address for the return parameter was specified.

ValueList

Property: ValueList as Integer

Description
Generate not more than this amount of enumeration-facets per type. Set to -1 for unlimited.

Errors
3000 The object is no longer valid.
3001 Invalid address for the return parameter was specified.

4.3.11 ElementList

See also

Properties
Count
Item

Methods
RemoveElement

Description
Element lists are used for different purposes during export and import of data. Depending on this purpose, different properties of ElementListItem are used.

It can hold
- a list of table names returned by a call to Application.GetDatabaseTables,
- a list of field names retuned by a call to Application.GetDatabaseImportElementList or Application.GetTextImportElementList,
- a field name filter list used in Application.ImportFromDatabase and Application.ImportFromText,
- a list of table names and counts for their rows and columns as returned by calls to GetExportElementList or
- a field name filter list used in Document.ExportToDatabase and Document.ExportToText.

Count

See also

Property: Count as long (read-only)

Description
Count of elements in this collection.

Item

See also

Method: Item(n as long) as ElementListItem

Description
Gets the element with the index n from this collection. The first item has index 1.

RemoveElement

See also

© 2007 Altova GmbH

Method: RemoveElement(Index as long)

Description
RemoveElement removes the element Index from the collection. The first Item has index 1.

4.3.12 ElementListItem

See also

Properties
Name

ElementKind

FieldCount
RecordCount

Description
An element in an ElementList. Usage of its properties depends on the purpose of the element list. For details see ElementList.

ElementKind

See also

Property: ElementKind as SPYXMLDataKind

Description
Specifies if a field should be imported as XML element (data value of spyXMLDataElement) or attribute (data value of spyXMLDataAttr).

FieldCount

See also

Property: FieldCount as long (read-only)

Description
Count of fields (i.e. columns) in the table described by this element. This property is only valid after a call to DocumentGetExportElementList .

Name

See also

Property: Name as String (read-only)

Description
Name of the element. This is either the name of a table or a field, depending on the purpose of th element list.

RecordCount

See also

Property: `RecordCount` as long (read-only)

Description
Count of records (i.e. rows) in the table described by this element. This property is only valid after a call to `DocumentGetExportElementList`.

4.3.13 ExportSettings

See also

Properties

`ElementList`

`EntitiesToText`

`ExportAllElements`
`SubLevelLimit`

`FromAttributes`
`FromSingleSubElements`
`FromTextValues`

`CreateKeys`
`IndependentPrimaryKey`

`Namespace`

`ExportCompleteXML`
`StartFromElement`

Description
`ExportSettings` contains options used during export of XML data to a database or text file. See Import and export of data for a general overview.

CreateKeys

See also

Property: `CreateKeys` as Boolean

Description
This property turns creation of keys (i.e. primary key and foreign key) on or off. Default is True.

ElementList

See also

Property: `ElementList` as `ElementList`

Description
Default is empty list. This list of elements defines which fields will be exported. To get the list of available fields use `DocumentGetExportElementList`. It is possible to prevent exporting columns by removing elements from this list with `ElementList.RemoveElement` before passing it to `DocumentExportToDatabase` or `DocumentExportToText`.

© 2007 Altova GmbH

EntitiesToText

See also

Property: `EntitiesToText` as Boolean

Description
Defines if XML entities should be converted to text or left as they are during export. Default is True.

ExportAllElements

See also

Property: `ExportAllElements` as Boolean

Description
If set to true, all elements in the document will be exported. If set to false, then `ExportSettings.SubLevelLimit` is used to restrict the number of sub levels to export. Default is true.

ExportCompleteXML

See also

Property: ExportCompleteXML as Boolean

Description
Defines whether the complete XML is exported or only the element specified by StartFromElement and its children. Default is True.

FromAttributes

See also

Property: `FromAttributes` as Boolean

Description
SetFromAttributes to false if no export data should be created from attributes. Default is True.

FromSingleSubElements

See also

Property: `FromSingleSubElements` as Boolean

Description
SetFromSingleSubElements to false if no export data should be created from elements. Default is True.

FromTextValues

See also

Property: `FromTextValues` as Boolean

Description
`SetFromTextValues` to false if no export data should be created from text values. Default is True.

IndependentPrimaryKey

See also

Property: `IndependentPrimaryKey` as Boolean

Description
Turns creation of independent primary key counter for every element on or off. If `ExportSettings.CreateKeys` is False, this property will be ignored. Default is True.

Namespace

See also

Property: `Namespace` as `SPYExportNamespace`

Description
The default setting removes all namespace prefixes from the element names. In some database formats the colon is not a legal character. Default is `spyNoNamespace`.

StartFromElement

See also

Property: StartFromElement as String

Description
Specifies the start element for the export. This property is only considered when ExportCompleteXML is false.

SubLevelLimit

See also

Property: `SubLevelLimit` as Integer

Description
Defines the number of sub levels to include for the export. Default is 0. This property is ignored if `ExportSettings.ExportAllElements` is true.

4.3.14 FileSelectionDlg

See also

Properties and Methods

Standard automation properties
Application
Parent

© 2007 Altova GmbH

Dialog properties
FullName

Acceptance or cancellation of action that caused event
DialogAction

Description
The dialog object allows you to receive information about an event and pass back information to the event handler in the same way as with a user dialog. Use the FileSelectionDlg.FullName to select or modify the file path and set the FileSelectionDlg.DialogAction property to cancel or agree with the action that caused the event.

Application

See also

Property: Application as Application (read-only)

Description
Access the XMLSpy application object.

Errors
2400 The object is no longer valid.
2401 Invalid address for the return parameter was specified.

DialogAction

Property: DialogAction as SPYDialogAction

Description
If you want your script to perform the file selection operation without any user interaction necessary, simulate user interaction by either setting the property to *spyDialogOK(0)* or *spyDialogCancel(1)*.
To allow your script to fill in the default values but let the user see and react on the dialog, use the value *spyDialogUserInput(2)*. If you receive a FileSelectionDlg object in an event handler, *spyDialogUserInput(2)* is not supported and will be interpreted as *spyDialogOK(0)*.

Errors
2400 The object is no longer valid.
2401 Invalid value for dialog action or invalid address for the return parameter was specified.

FullName

Property: FullName as String

Description
Access the full path of the file the gets selected by the dialog. Most events that pass a FileSelectionDlg object to you allow you modify this value and thus influence the action that caused the event (e.g. load or save to a different location).

Errors
2400 The object is no longer valid.
2401 Invalid address for the return parameter was specified.

Parent

See also

Property: Parent as Dialogs (read-only)

Description
Access the parent of the object.

Errors
2400 The object is no longer valid.
2401 Invalid address for the return parameter was specified.

4.3.15 FindInFilesDlg

See also

Properties and Methods

Standard automation properties
Application
Parent

Find
RegularExpression
Replace
DoReplace
ReplaceOnDisk
MatchWholeWord
MatchCase
SearchLocation
StartFolder
IncludeSubfolders
SearchInProjectFilesDoExternal
FileExtension
AdvancedXMLSearch
XMLElementNames
XMLElementContents
XMLAttributeNames
XMLAttributeContents
XMLComments
XMLCData
XMLPI
XMLRest
ShowResult

Description
Use this object to configure the search (or replacement) for strings in files. The method FindInFiles expects a FindInFilesDlg as parameter.

AdvancedXMLSearch

Property: AdvancedXMLSearch as Boolean

Description
Specifies if the XML search properties (XMLElementNames, XMLElementContents,

© 2007 Altova GmbH

XMLAttributeNames, XMLAttributeContents, XMLComments, XMLCData, XMLPI and XMLRest) are considered. The default is false.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

Application

Property: `Application` as `Application` (read-only)

Description
Access the XMLSpy application object.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

DoReplace

Property: DoReplace as Boolean

Description
Specifies if the matched string is replaced by the string defined in Replace. The default is false.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

FileExtension

Property: FileExtension as String

Description
Specifies the file filter of the files that should be considered during the search. Multiple file filters must be delimited with a semicolon (eg: *.xml;*.dtd;a*.xsd). Use the wildcards * and ? to define the file filter.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

Find

Property: Find as String

Description
Specifies the string to search for.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

IncludeSubfolders

Property: IncludeSubfolders as Boolean

Description
Specifies if subfolders are searched too. The default is true.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

MatchCase

Property: MatchCase as Boolean

Description
Specifies if the search is case sensitive. The default is true.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

MatchWholeWord

Property: MatchWholeWord as Boolean

Description
Specifies whether the whole word or just a part of it must match. The default is false.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

Parent

Property: `Parent` as `Dialogs` (read-only)

Description
Access the parent of the object.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

RegularExpression

Property: RegularExpression as Boolean

Description
Specifies if Find contains a regular expression. The default is false.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

© 2007 Altova GmbH

Replace

Property: Replace as String

Description
Specifies the replacement string. The matched string is only replaced if DoReplace is set true.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

ReplaceOnDisk

Property: ReplaceOnDisk as Boolean

Description
Specifies if the replacement is done directly on disk. The modified file is not opened. The default is false.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

SearchInProjectFilesDoExternal

Property: SearchInProjectFilesDoExternal as Boolean

Description
Specifies if the external folders in the open project are searched, when a project search is performed. The default is false.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

SearchLocation

Property: SearchLocation as SPYFindInFilesSearchLocation

Description
Specifies the location of the search. The default is spyFindInFiles_Documents.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

ShowResult

Property: ShowResult as Boolean

Description
Specifies if the result is displayed in the Find in Files output window. The default is false.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

StartFolder

Property: StartFolder as String

Description
Specifies the folder where the disk search starts.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

XMLAttributeContents

Property: XMLAttributeContents as Boolean

Description
Specifies if attribute contents are searched when AdvancedXMLSearch is true. The default is true.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

XMLAttributeNames

Property: XMLAttributeNames as Boolean

Description
Specifies if attribute names are searched when AdvancedXMLSearch is true. The default is true.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

XMLCData

Property: XMLCData as Boolean

Description
Specifies if CData tags are searched when AdvancedXMLSearch is true. The default is true.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

XMLComments

Property: XMLComments as Boolean

Description
Specifies if comments are searched when AdvancedXMLSearch is true. The default is true.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

© 2007 Altova GmbH

XMLElementContents

Property: XMLElementContents as Boolean

Description
Specifies if element contents are searched when AdvancedXMLSearch is true. The default is true.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

XMLElementNames

Property: XMLElementNames as Boolean

Description
Specifies if element names are searched when AdvancedXMLSearch is true. The default is true.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

XMLPI

Property: XMLPI as Boolean

Description
Specifies if XML processing instructions are searched when AdvancedXMLSearch is true. The default is true.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

XMLRest

Property: XMLRest as Boolean

Description
Specifies if the rest of the XML (which is not covered by the other XML search properties) is searched when AdvancedXMLSearch is true. The default is true.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

4.3.16 FindInFilesResult

See also

Properties and Methods

Standard automation properties

Application
Parent

Count
Item

Path
Document

Description
This object represents a file that matched the search criteria. It contains a list of FindInFilesResultMatch objects that describe the matching position.

Application

Property: `Application` as `Application` (read-only)

Description
Access the XMLSpy application object.

Errors
3700 The object is no longer valid.
3701 Invalid address for the return parameter was specified.

Count

Property: `Count` as long (read-only)

Description
Count of elements in this collection.

Document

Property: Path as Document (read-only)

Description
This property returns the Document object if the matched file is already open in XMLSpy.

Errors
3700 The object is no longer valid.
3701 Invalid address for the return parameter was specified.

Item

Method: `Item(n` as long) as FindInFilesResultMatch

Description
Gets the element with the index `n` from this collection. The first item has index 1.

Parent

Property: `Parent` as FindInFilesResults (read-only)

Description

© 2007 Altova GmbH

Access the parent of the object.

Errors
3700 The object is no longer valid.
3701 Invalid address for the return parameter was specified.

Path

Property: Path as String (read-only)

Description
Returns the path of the file that matched the search criteria.

Errors
3700 The object is no longer valid.
3701 Invalid address for the return parameter was specified.

4.3.17 FindInFilesResultMatch

See also

Properties and Methods

Standard automation properties
Application
Parent

Line
Position
Length
LineText
Replaced

Description
Contains the exact position in the file of the matched string.

Application

Property: `Application` as `Application` (read-only)

Description
Access the XMLSpy application object.

Errors
3800 The object is no longer valid.
3801 Invalid address for the return parameter was specified.

Length

Property: Length as Long (read-only)

Description
Returns the length of the matched string.

Errors
3800 The object is no longer valid.
3801 Invalid address for the return parameter was specified.

Line

Property: Line as Long (read-only)

Description
Returns the line number of the match. The line numbering starts with 0.

Errors
3800 The object is no longer valid.
3801 Invalid address for the return parameter was specified.

LineText

Property: LineText as String (read-only)

Description
Returns the text of the line.

Errors
3800 The object is no longer valid.
3801 Invalid address for the return parameter was specified.

Parent

Property: `Parent` as FindInFilesResult (read-only)

Description
Access the parent of the object.

Errors
3800 The object is no longer valid.
3801 Invalid address for the return parameter was specified.

Position

Property: Position as Long (read-only)

Description
Returns the start position of the match in the line. The position numbering starts with 0.

Errors
3800 The object is no longer valid.
3801 Invalid address for the return parameter was specified.

© 2007 Altova GmbH

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.

4.3.18 FindInFilesResults

See also

Properties and Methods

Standard automation properties
Application
Parent

Count
Item

Description
This is the result of the FindInFiles method. It is a list of FindInFilesResult objects.

Application

Property: `Application` as `Application` (read-only)

Description
Access the XMLSpy application object.

Errors
3600 The object is no longer valid.
3601 Invalid address for the return parameter was specified.

Count

Property: `Count` as long (read-only)

Description
Count of elements in this collection.

Item

Method: `Item(n` as long) as FindInFilesResult

Description
Gets the element with the index `n` from this collection. The first item has index 1.

Parent

Property: `Parent` as Application (read-only)

Description
Access the parent of the object.

Errors
3600 The object is no longer valid.
3601 Invalid address for the return parameter was specified.

4.3.19 GenerateSampleXMLDlg

See also

Properties and Methods

Standard automation properties
Application
Parent

NonMandatoryAttributes
RepeatCount
FillAttributesWithSampleData
FillElementsWithSampleData
ContentOfNillableElementsIsNonMandatory
TryToUseNonAbstractTypes
Optimization
SchemaOrDTDAssignment
LocalNameOfRootElement
NamespaceURIOfRootElement
OptionsDialogAction

Properties that are no longer supported
NonMandatoryElements - obsolete
TakeFirstChoice - obsolete
FillWithSampleData - obsolete

Description
Used to set the parameters for the generation of sample XML instances based on a W3C schema or DTD.

Application

Property: `Application` as Application (read-only)

Description
Access the XMLSpy application object.

Errors
2200 The object is no longer valid.
2201 Invalid address for the return parameter was specified.

© 2007 Altova GmbH

Parent

Property: `Parent` as `Dialogs` (read-only)

Description
Access the parent of the object.

Errors
2200 The object is no longer valid.
2201 Invalid address for the return parameter was specified.

NonMandatoryAttributes

Property: `NonMandatoryAttributes` as Boolean

Description
If true attributes which are not mandatory are created in the sample XML instance file.

Errors
2200 The object is no longer valid.
2201 Invalid address for the return parameter was specified.

NonMandatoryElements - obsolete

Property: `NonMandatoryElements` as Boolean

Description
Do no longer use this property. Use `Optimization`, instead.

Errors
0001 The property is no longer accessible.

TakeFirstChoice - obsolete

Property: `TakeFirstChoice` as Boolean

Description
Do no longer use this property.

Errors
0001 The property is no longer accessible.

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.

FillWithSampleData - obsolete

Property: `FillWithSampleData` as Boolean

Description
Do no longer access this property. Use `FillAttributesWithSampleData` and `FillElementsWithSampleData`, instead.

Errors
0001 The property is no longer accessible.

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.

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.

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.

TryToUseNonAbstractTypes

Property: `TryToUseNonAbstractTypes` as Boolean

Description
If true, tries to use a non-abstract type for xsi:type, if element has an abstract type.

Errors
2200 The object is no longer valid.
2201 Invalid address for the return parameter was specified.

© 2007 Altova GmbH

Optimization

Property: `Optimization` as `SPYSampleXMLGenerationOptimization`

Description
Specifies which elements will be generated.

Errors
2200 The object is no longer valid.
2201 Invalid address for the return parameter was specified.

SchemaOrDTDAssignment

Property: `SchemaOrDTDAssignment` as `SPYSampleXMLGenerationSchemaOrDTDAssignment`

Description
Specifies in which way a reference to the related schema or DTD - which is this document - will be generated into the sample XML.

Errors
2200 The object is no longer valid.
2201 Invalid address for the return parameter was specified.

LocalNameOfRootElement

Property: `LocalNameOfRootElement` as String

Description
Specifies the local name of the root element for the generated sample XML.

Errors
2200 The object is no longer valid.
2201 Invalid address for the return parameter was specified.

NamespaceURIOfRootElement

Property: `NamespaceURIOfRootElement` as String

Description
Specifies the namespace URI of the root element for the generated sample XML.

Errors
2200 The object is no longer valid.
2201 Invalid address for the return parameter was specified.

OptionsDialogAction

Property: `OptionsDialogAction` as `SPYDialogAction`

Description
To allow your script to fill in the default values and let the user see and react on the dialog, set this property to the value *spyDialogUserInput(2)*. If you want your script to define all the options in the schema documentation dialog without any user interaction necessary, use *spyDialogOK(0)*. Default is *spyDialogOK.*

Errors

2200 The object is no longer valid.
2201 Invalid value has been used to set the property.
Invalid address for the return parameter was specified.

4.3.20 GridView

See also

Methods

Deselect
Select

SetFocus

Properties

CurrentFocus

IsVisible

Description

GridView Class

Events

OnBeforeDrag

See also

Event: OnBeforeDrag() as Boolean

XMLSpy scripting environment - VBScript:

```
Function On_BeforeDrag()
	'On_BeforeStartEditing=False 'to prohibit dragging
End Function
```

XMLSpy scripting environment - JScript:

```
function On_BeforeDrag()
{
	//return false;/*to prohibit dragging*/
}
```

XMLSpy IDE Plugin:

```
IXMLSpyPlugIn.OnEvent (4, ...)	//nEventID=4
```

Description

This event gets fired on an attempt to drag an XMLData element on the grid view. Return *false* to prevent dragging the data element to a different position.

OnBeforeDrop

See also

Event: OnBeforeDrop(*objXMLData* as XMLData) as Boolean

© 2007 Altova GmbH

XMLSpy scripting environment - VBScript:

```
Function On_BeforeDrop(objXMLData)
	'On_BeforeStartEditing=False 'to prohibit dropping
End Function
```

XMLSpy scripting environment - JScript:

```
function On_BeforeDrop(objXMLData)
{
	//return false;/*to prohibit dropping*/
}
```

XMLSpy IDE Plugin:

```
IXMLSpyPlugIn.OnEvent (5, ...)	//nEventID=5
```

Description

This event gets fired on an attempt to drop a previously dragged XMLData element on the grid view. Return *false* to prevent the data element to be moved from its original position to the drop destination position.

OnBeforeStartEditing

See also

Event: `OnBeforeStartEditing(objXMLData as XMLData, bEditingName as Boolean) as Boolean`

XMLSpy scripting environment - VBScript:

```
Function On_BeforeStartEditing(objXMLData, bEditingName)
	'On_BeforeStartEditing=False 'to prohibit editing the field
End Function
```

XMLSpy scripting environment - JScript:

```
function On_BeforeStartEditing(objXMLData, bEditingName)
{
	//return false;/*to prohibit editing the field*/
}
```

XMLSpy IDE Plugin:

```
IXMLSpyPlugIn.OnEvent (1, ...)	//nEventID=1
```

Description

This event gets fired before the editing mode for a grid cell gets entered. If the parameter *bEditingName* is true, the name part of the element will be edited, it its value is false, the value part will be edited.

OnEditingFinished

See also

Event: `OnEditingFinished(objXMLData as XMLData, bEditingName as Boolean)`

XMLSpy scripting environment - VBScript:

```
Function On_EditingFinished(objXMLData, bEditingName)
End Function
```

XMLSpy scripting environment - JScript:

```
function On_EditingFinished(objXMLData, bEditingName)
{
}
```

XMLSpy IDE Plugin:

```
IXMLSpyPlugIn.OnEvent (2, ...)    //nEventId=2
```

Description

This event gets fired when the editing mode of a grid cell gets left. The parameter *bEditingName* specifies if the name part of the element has been edited.

OnFocusChanged

See also

Event: `OnFocusChanged(objXMLData as XMLData, bSetFocus as Boolean, bEditingName as Boolean)`

XMLSpy scripting environment - VBScript:

```
Function On_FocusChanged(objXMLData, bSetFocus, bEditingName)
End Function
```

XMLSpy scripting environment - JScript:

```
function On_FocusChanged(objXMLData, bSetFocus, bEditingName)
{
}
```

XMLSpy IDE Plugin:

```
IXMLSpyPlugIn.OnEvent (3, ...)    //nEventId=3
```

Description

This event gets fired whenever a grid cell receives or looses the cursor focus. If the parameter *bEditingName* is *true*, focus of the name part of the grid element has changed. Otherwise, focus of the value part has changed.

CurrentFocus

See also

Property: `CurrentFocus as XMLData`

Description

Holds the XML element with the current focus. This property is read-only.

Deselect

See also

© 2007 Altova GmbH

Method: Deselect(*pData* as XMLData)

Description
Deselects the element pData in the grid view.

IsVisible

See also

Property: IsVisible as Boolean

Description
True if the grid view is the active view of the document. This property is read-only.

Select

See also

Method: Select (*pData* as XMLData)

Description
Selects the XML element pData in the grid view.

SetFocus

See also

Method: SetFocus (*pFocusData* as XMLData)

Description
Sets the focus to the element pFocusData in the grid view.

4.3.21 SchemaDocumentationDlg

See also

Properties and Methods

Standard automation properties
Application
Parent

Interaction and visibility properties
OutputFile
OutputFileDialogAction
OptionsDialogAction
ShowProgressBar
ShowResult

Document generation options and methods
OutputFormat
EmbedDiagrams
DiagramFormat
MultipleOutputFiles

IncludeAll

IncludeIndex
IncludeGlobalAttributes
IncludeGlobalElements
IncludeLocalAttributes
IncludeLocalElements
IncludeGroups
IncludeComplexTypes
IncludeSimpleTypes
IncludeAttributeGroups
IncludeRedefines
IncludeReferencedSchemas

AllDetails
ShowDiagram
ShowNamespace
ShowType
ShowChildren
ShowUsedBy
ShowProperties
ShowSingleFacets
ShowPatterns
ShowEnumerations
ShowAttributes
ShowIdentityConstraints
ShowAnnotations
ShowSourceCode

Description
This object combines all options for schema document generation as they are available through user interface dialog boxes in XMLSpy. The document generation options are initialized with the values used during the last generation of schema documentation. However, before using the object you have to set the OutputFile property to a valid file path. Use OptionsDialogAction, OutputFileDialogAction and ShowProgressBar to specify the level of user interaction desired. You can use IncludeAll and AllDetails to set whole option groups at once or the individual properties to operate on a finer granularity.

AllDetails

See also

Method: AllDetails (i_bDetailsOn as Boolean)

Description
Use this method to turn all details options on or off.

Errors
2900 The object is no longer valid.

Application

See also

 © 2007 Altova GmbH

Property: `Application` as `Application` (read-only)

Description
Access the XMLSpy application object.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

DiagramFormat

See also

Property: `DiagramFormat` as SPYImageKind

Description
Set this property to true, to specify the diagram image type. This property is not available for HTML documentation. The property is initialized with the value used during the last call to DocumentGenerateSchemaDocumentation. The default for the first run is PNG.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

EmbedDiagrams

See also

Property: `EmbedDiagrams` as Boolean

Description
Set this property to true, to embed the diagrams in the generated document. This property is not available for HTML documentation. The property is initialized with the value used during the last call to DocumentGenerateSchemaDocumentation. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

IncludeAll

See also

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.

IncludeAttributeGroups

See also

Property: `IncludeAttributeGroups` as Boolean

Description
Set this property to true, to include attribute groups in the schema documentation. The property is initialized with the value used during the last call to DocumentGenerateSchemaDocumentation. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

IncludeComplexTypes

See also

Property: `IncludeComplexTypes` as Boolean

Description
Set this property to true, to include complex types in the schema documentation. The property is initialized with the value used during the last call to DocumentGenerateSchemaDocumentation. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

IncludeGlobalAttributes

See also

Property: `IncludeGlobalAttributes` as Boolean

Description
Set this property to true, to include global attributes in the schema documentation. The property is initialized with the value used during the last call to DocumentGenerateSchemaDocumentation. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

IncludeGlobalElements

See also

Property: `IncludeGlobalElements` as Boolean

Description
Set this property to true, to include global elements in the schema documentation. The property is initialized with the value used during the last call to

© 2007 Altova GmbH

DocumentGenerateSchemaDocumentation. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

IncludeGroups

See also

Property: `IncludeGroups` as Boolean

Description
Set this property to true, to include groups in the schema documentation. The property is initialized with the value used during the last call to DocumentGenerateSchemaDocumentation. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

IncludeIndex

See also

Property: `IncludeIndex` as Boolean

Description
Set this property to true, to include an index in the schema documentation. The property is initialized with the value used during the last call to DocumentGenerateSchemaDocumentation. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

IncludeLocalAttributes

See also

Property: `IncludeLocalAttributes` as Boolean

Description
Set this property to true, to include local attributes in the schema documentation. The property is initialized with the value used during the last call to DocumentGenerateSchemaDocumentation. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

IncludeLocalElements

See also

Property: `IncludeLocalElements` as Boolean

Description
Set this property to true, to include local elements in the schema documentation. The property is initialized with the value used during the last call to DocumentGenerateSchemaDocumentation. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

IncludeRedefines

See also

Property: `IncludeRedefines` as Boolean

Description
Set this property to true, to include redefines in the schema documentation. The property is initialized with the value used during the last call to DocumentGenerateSchemaDocumentation. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

IncludeReferencedSchemas

See also

Property: `IncludeReferencedSchemas` as Boolean

Description
Set this property to true, to include referenced schemas in the schema documentation. The property is initialized with the value used during the last call to DocumentGenerateSchemaDocumentation. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

IncludeSimpleTypes

See also

Property: `IncludeSimpleTypes` as Boolean

Description
Set this property to true, to include simple types in the schema documentation. The property is initialized with the value used during the last call to DocumentGenerateSchemaDocumentation. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

© 2007 Altova GmbH

MultipleOutputFiles

See also

Property: `MultipleOutputFiles` as Boolean

Description
Set this property to true , to split the documentation files. The property is initialized with the value used during the last call to DocumentGenerateSchemaDocumentation. The default for the first run is false.

Errors

2900	The object is no longer valid.
2901	Invalid value has been used to set the property. Invalid address for the return parameter was specified.

OptionsDialogAction

See also

Property: `OptionsDialogAction` as SPYDialogAction

Description
To allow your script to fill in the default values and let the user see and react on the dialog, set this property to the value *spyDialogUserInput(2)*. If you want your script to define all the options in the schema documentation dialog without any user interaction necessary, use *spyDialogOK(0)*. Default is *spyDialogOK.*

Errors

2900	The object is no longer valid.
2901	Invalid value has been used to set the property. Invalid address for the return parameter was specified.

OutputFile

See also

Property: `OutputFile` as String

Description
Full path and name of the file that will contain the generated documentation. In case of HTML output, additional '.png' files will be generated based on this filename. The default value for this property is an empty string and needs to be replaced before using this object in a call to DocumentGenerateSchemaDocumentation.

Errors

2900	The object is no longer valid.
2901	Invalid address for the return parameter was specified.

OutputFileDialogAction

See also

Property: `OutputFileDialogAction` as SPYDialogAction

Description
To allow the user to select the output file with a file selection dialog, set this property to *spyDialogUserInput(2)*. If the value stored in OutputFile should be taken and no user interaction should occur, use *spyDialogOK(0)*. Default is *spyDialogOK*.

Errors
2900 The object is no longer valid.
2901 Invalid value has been used to set the property.
Invalid address for the return parameter was specified.

OutputFormat

See also

Property: `OutputFormat` as `SPYSchemaDocumentationFormat`

Description
Defines the kind of documentation that will be generated: HTML (value=0), MS-Word (value=1), or RTF (value=2). The property gets initialized with the value used during the last call to `DocumentGenerateSchemaDocumentation`. The default for the first run is HTML.

Errors
2900 The object is no longer valid.
2901 Invalid value has been used to set the property.
Invalid address for the return parameter was specified.

Parent

See also

Property: `Parent` as `Dialogs` (read-only)

Description
Access the parent of the object.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

ShowAnnotations

See also

Property: `ShowAnnotations` as Boolean

Description
Set this property to true, to show the annotations to a type definition in the schema documentation. The property is initialized with the value used during the last call to `DocumentGenerateSchemaDocumentation`. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

© 2007 Altova GmbH

ShowAttributes

See also

Property: ShowAttributes as Boolean

Description
Set this property to true , to show the type definitions attributes in the schema documentation. The property is initialized with the value used during the last call to DocumentGenerateSchemaDocumentation. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

ShowChildren

See also

Property: ShowChildren as Boolean

Description
Set this property to true , to show the children of a type definition as links in the schema documentation. The property is initialized with the value used during the last call to DocumentGenerateSchemaDocumentation. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

ShowDiagram

See also

Property: ShowDiagram as Boolean

Description
Set this property to true , to show type definitions as diagrams in the schema documentation. The property is initialized with the value used during the last call to DocumentGenerateSchemaDocumentation. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

ShowEnumerations

See also

Property: ShowEnumerations as Boolean

Description
Set this property to true , to show the enumerations contained in a type definition in the schema

documentation. The property is initialized with the value used during the last call to DocumentGenerateSchemaDocumentation. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

ShowIdentityConstraints

See also

Property: ShowIdentityConstraints as Boolean

Description
Set this property to true, to show a type definitions identity constraints in the schema documentation. The property is initialized with the value used during the last call to DocumentGenerateSchemaDocumentation. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

ShowNamespace

See also

Property: ShowNamespace as Boolean

Description
Set this property to true, to show the namespace of type definitions in the schema documentation. The property is initialized with the value used during the last call to DocumentGenerateSchemaDocumentation. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

ShowPatterns

See also

Property: ShowPatterns as Boolean

Description
Set this property to true, to show the patterns of a type definition in the schema documentation. The property is initialized with the value used during the last call to DocumentGenerateSchemaDocumentation. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

© 2007 Altova GmbH

ShowProgressBar

See also

Property: ShowProgressBar as Boolean

Description
Set this property to true, to make the window showing the document generation progress visible. Use false, to hide it. Default is false.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

ShowProperties

See also

Property: ShowProperties as Boolean

Description
Set this property to true, to show the type definition properties in the schema documentation. The property is initialized with the value used during the last call to DocumentGenerateSchemaDocumentation. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

ShowResult

See also

Property: ShowResult as Boolean

Description
Set this property to true, to automatically open the resulting document when generation was successful. HTML documentation will be opened in XMLSpy. To show Word documentation, MS-Word will be started. The property gets initialized with the value used during the last call to DocumentGenerateSchemaDocumentation. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

ShowSingleFacets

See also

Property: ShowSingleFacets as Boolean

Description
Set this property to true, to show the facets of a type definition in the schema documentation. The property is initialized with the value used during the last call to

Document.GenerateSchemaDocumentation. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

ShowSourceCode

See also

Property: ShowSourceCode as Boolean

Description
Set this property to true, to show the XML source code for type definitions in the schema documentation. The property is initialized with the value used during the last call to Document.GenerateSchemaDocumentation. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

ShowType

See also

Property: ShowType as Boolean

Description
Set this property to true, to show the type of type definitions in the schema documentation. The property is initialized with the value used during the last call to Document.GenerateSchemaDocumentation. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

ShowUsedBy

See also

Property: ShowUsedBy as Boolean

Description
Set this property to true, to show the used-by relation for type definitions in the schema documentation. The property is initialized with the value used during the last call to Document.GenerateSchemaDocumentation. 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.

© 2007 Altova GmbH

4.3.22 SpyProject

See also

Methods
CloseProject
SaveProject
SaveProjectAs

Properties
RootItems
ProjectFile

Description
SpyProject Class

CloseProject

See also

Declaration: `CloseProject(`*`bDiscardChanges`* as Boolean, *`bCloseFiles`* as Boolean, *`bDialog`* as Boolean)

Parameters
`bDiscardChanges`
Set `bDiscardChanges` to FALSE if you want to save the changes of the open project files and the project.

`bCloseFiles`
Set `bCloseFiles` to TRUE to close all open project files.

`bDialog`
Show dialogs for user input.

Description
`CloseProject` closes the current project.

ProjectFile

See also

Declaration: `ProjectFile` as String

Description
Path and filename of the project.

RootItems

See also

Declaration: `RootItems` as SpyProjectItems

Description
Root level of collection of project items.

SaveProject

See also

Declaration: SaveProject

Description
SaveProject saves the current project.

SaveProjectAs

See also

Declaration: SaveProjectAs (strPath as String, bDialog as Boolean)

Parameters
strPath
Full path with file name of new project file.

bDialog
If bDialog is TRUE, a file-dialog will be displayed.

Description
SaveProjectAs stores the project data into a new location.

4.3.23 SpyProjectItem

See also

Methods
Open

Properties
ChildItems
ParentItem
FileExtensions
ItemType
Name
Path
ValidateWith
XMLForXSLTransformation
XSLForXMLTransformation
XSLTransformationFileExtension
XSLTransformationFolder

Description
SpyProjectItem Class

ChildItems

See also

Declaration: ChildItems as SpyProjectItems

Description

© 2007 Altova GmbH

If the item is a folder, ChildItems is the collection of the folder content.

FileExtensions

See also

Declaration: FileExtensions as String

Description
Used to set the file extensions if the project item is a folder.

ItemType

See also

Declaration: ItemType as SPYProjectItemTypes

Description
This property is read-only.

Name

See also

Declaration: Name as String

Description
Name of the project item. This property is read-only.

Open

See also

Declaration: Open as Document

Return Value
The project item opened as document.

Description
Opens the project item.

ParentItem

See also

Declaration: ParentItem as SpyProjectItem

Description
Parent item of the current project item. Can be NULL (Nothing) if the project item is a top-level item.

Path

See also

Declaration: `Path` as String

Description
Path of project item. This property is read-only.

ValidateWith

See also

Declaration: `ValidateWith` as String

Description
Used to set the schema/DTD for validation.

XMLForXSLTransformation

See also

Declaration: `XMLForXSLTransformation` as String

Description
Used to set the XML for XSL transformation.

XSLForXMLTransformation

See also

Declaration: `XSLForXMLTransformation` as String

Description
Used to set the XSL for XML transformation.

XSLTransformationFileExtension

See also

Declaration: `XSLTransformationFileExtension` as String

Description
Used to set the file extension for XSL transformation output files.

XSLTransformationFolder

See also

Declaration: `XSLTransformationFolder` as String

Description
Used to set the destination folder for XSL transformation output files.

© 2007 Altova GmbH

4.3.24 SpyProjectItems

See also

Methods
AddFile
AddFolder
AddURL
RemoveItem

Properties
Count
Item

Description
SpyProjectItems Class

AddFile

See also

Declaration: AddFile (*strPath* as String)

Parameters

strPath
Full path with file name of new project item

Description
The method adds a new file to the collection of project items.

AddFolder

See also

Declaration: AddFolder (*strName* as String)

Parameters

strName
Name of the new folder.

Description
The method AddFolder adds a folder with the name strName to the collection of project items.

AddURL

See also

Declaration: AddURL (*strURL* as String, *nURLType* as SPYURLTypes, *strUser* as String, *strPassword* as String, *bSave* as Boolean)

Description

strURL
URL to open as document.

nURLType
Type of document to open. Set to -1 for auto detection.

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.

bSave
Save user and password information.

Description
The method adds an URL item to the project collection.

Count

See also

Declaration: Count as long

Description
This property gets the count of project items in the collection. The property is read-only.

Item

See also

Declaration: Item (*n* as long) as SpyProjectItem

Description
Retrieves the n-th element of the collection of project items. The first item has index 1.

RemoveItem

See also

Declaration: RemoveItem (*pItem* as SpyProjectItem)

Description
RemoveItem deletes the item pItem from the collection of project items.

4.3.25 TextImportExportSettings

See also

Properties for import only
ImportFile

© 2007 Altova GmbH

Properties for export only
DestinationFolder
FileExtension
CommentIncluded
RemoveDelimiter
RemoveNewline

Properties for import and export
HeaderRow
FieldDelimiter
EnclosingCharacter
Encoding
EncodingByteOrder

Description
TextImportExportSettings contains options common to text import and export functions.

CommentIncluded

See also

Property: CommentIncluded as Boolean

Description
This property tells whether additional comments are added to the generated text file. Default is true. This property is used only when exporting to text files.

DestinationFolder

See also

Property: `DestinationFolder` as String

Description
The property `DestinationFolder` sets the folder where the created files are saved during text export.

EnclosingCharacter

See also

Property: `EnclosingCharacter` as `SPYTextEnclosing`

Description
This property defines the character that encloses all field values for import and export. Default is `spyNoEnclosing`.

Encoding

See also

Property: `Encoding` as String

Description
The property `Encoding` sets the character encoding for the text files for importing and

exporting.

EncodingByteOrder

See also

Property: `EncodingByteOrder` as `SPYEncodingByteOrder`

Description
The property `EncodingByteOrder` sets the byte order for Unicode characters. Default is `spyNONE`.

FieldDelimiter

See also

Property: `FieldDelimiter` as `SPYTextDelimiters`

Description
The property `FieldDelimiter` defines the delimiter between the fields during import and export. Default is `spyTabulator`.

FileExtension

See also

Property: `FileExtension` as String

Description
This property sets the file extension for files created on text export.

HeaderRow

See also

Property: `HeaderRow` as Boolean

Description
The property `HeaderRow` is used during import and export. Set `HeaderRow` true on import, if the first line of the text file contains the names of the columns. Set `HeaderRow` true on export, if the first line in the created text files should contain the name of the columns. Default value is true.

ImportFile

See also

Property: `ImportFile` as String

Description
This property is used to set the text file for import. The string has to be a full qualified path. See also Import and Export.

© 2007 Altova GmbH

RemoveDelimiter

See also

Property: RemoveDelimiter as Boolean

Description
The property RemoveDelimiter defines whether characters in the text that are equal to the delimiter character are removed. Default is false. This property is used only when exporting to text files.

RemoveNewline

See also

Property: RemoveNewline as Boolean

Description
The property RemoveNewline defines whether newline characters in the text are removed. Default is false. This property is used only when exporting to text files.

4.3.26 TextView

See also

Properties and Methods

Application
Parent

LineFromPosition
PositionFromLine
LineLength
SelText
GetRangeText
ReplaceText
MoveCaret
GoToLineChar
SelectText
SelectionStart
SelectionEnd
Text
LineCount
Length

Description

Events

OnBeforeShowSuggestions

See also

Event: OnBeforeShowSuggestions() as Boolean

Description
This event gets fired before a suggestion window is shown. The Document property Suggestions contains a string array that is recommended to the user. It is possible to modify the displayed recommendations during this event. Before doing so you have to assign an empty array to the Suggestions property. The best location for this is the OnDocumentOpened event. To prevent the suggestion window to show up return false and true to continue its display.

Examples
Given below are examples of how this event can be scripted.

XMLSpy scripting environment - VBScript:
```
Function On_BeforeShowSuggestions()
End Function
```

XMLSpy scripting environment - JScript:
```
function On_BeforeShowSuggestions()
{
}
```

XMLSpy IDE Plugin:
```
IXMLSpyPlugIn.OnEvent (33, ...)  //nEventId=33
```

OnChar

See also

Event: OnChar(nChar as Long, bExistSuggestion as Boolean) as Boolean

Description
This event gets fired on each key stroke. The parameter nChar is the key that was pressed and bExistSuggestions tells whether a XMLSpy generated suggestions window is displayed after this key. The Document property Suggestions contains a string array that is recommended to the user. It is possible to modify the displayed recommendations during this event. Before doing so you have to assign an empty array to the Suggestions property. The best location for this is the OnDocumentOpened event. To prevent the suggestion window to show up return false and true to continue its display.
It is also possible to create a new suggestions window when none is provided by XMLSpy. Set the Document property Suggestions to a string array with your recommendations and return true.
This event is fired before the OnBeforeShowSuggestions event. If you prevent to show the suggestion window by returning false then OnBeforeShowSuggestions is not fired.

Examples
Given below are examples of how this event can be scripted.

XMLSpy scripting environment - VBScript:
```
Function On_Char(nChar, bExistSuggestions )
End Function
```

XMLSpy scripting environment - JScript:
```
function On_Char(nChar, bExistSuggestions )
{
}
```

 © 2007 Altova GmbH

XMLSpy IDE Plugin:
```
IXMLSpyPlugIn.OnEvent (35, ...)   //nEventID=35
```

Application

Property: Application as Application (read-only)

Description
Access the XMLSpy application object.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

GetRangeText

Method: GetRangeText(nStart as Long, nEnd as Long) as String

Description
Returns the text in the specified range.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

GoToLineChar

Method: GoToLineChar(nLine as Long, nChar as Long)

Description
Moves the caret to the specified line and character position.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

Length

Property: Length as Long

Description
Returns the character count of the document.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

LineCount

Property: LineCount as Long

Description
Returns the number of lines in the document.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

LineFromPosition

Method: LineFromPosition(nCharPos as Long) as Long

Description
Returns the line number of the character position.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

LineLength

Method: LineLength(nLine as Long) as Long

Description
Returns the length of the line.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

MoveCaret

Method: MoveCaret(nDiff as Long)

Description
Moves the caret nDiff characters.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

Parent

Property: `Parent` as Document (read-only)

Description
Access the parent of the object.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

PositionFromLine

Method: PositionFromLine(nLine as Long) as Long

Description
Returns the start position of the line.

© 2007 Altova GmbH

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

ReplaceText

Method: ReplaceText(nPosFrom as Long, nPosTill as Long, sText as String)

Description
Replaces the text in the specified range.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

SelectionEnd

Property: SelectionEnd as Long

Description
Returns/sets the text selection end position.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

SelectionStart

Property: SelectionStart as Long

Description
Returns/sets the text selection start position.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

SelectText

Method: SelectText(nPosFrom as Long, nPosTill as Long)

Description
Selects the text in the specified range.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

SelText

Property: SelText as String

Description
Returns/sets the selected text.

Errors

3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

Text

Property: Text as String

Description
Returns/sets the document text.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

4.3.27 WSDLDocumentationDlg

See also

Properties and Methods

Standard automation properties
Application
Parent

Interaction and visibility properties
OutputFile
OutputFileDialogAction
OptionsDialogAction
ShowProgressBar
ShowResult

Document generation options and methods
OutputFormat
EmbedDiagrams
DiagramFormat
MultipleOutputFiles

IncludeAll
IncludeBinding
IncludeImportedWSDLFiles
IncludeMessages
IncludeOverview
IncludePortType
IncludeService
IncludeTypes

AllDetails
ShowBindingDiagram
ShowExtensibility
ShowMessageParts
ShowPort
ShowPortTypeDiagram
ShowPortTypeOperations
ShowServiceDiagram
ShowSourceCode
ShowTypesDiagram
ShowUsedBy

© 2007 Altova GmbH

Description
This object combines all options for WSDL document generation as they are available through user interface dialog boxes in XMLSpy. The document generation options are initialized with the values used during the last generation of WSDL documentation. However, before using the object you have to set the `OutputFile` property to a valid file path. Use `OptionsDialogAction`, `OutputFileDialogAction` and `ShowProgressBar` to specify the level of user interaction desired. You can use `FolderAll` and `AllDetails` to set whole option groups at once or the individual properties to operate on a finer granularity.

AllDetails

See also

Method: `AllDetails` (`i_bDetailsOn` as Boolean)

Description
Use this method to turn all details options on or off.

Errors
3900 The object is no longer valid.

Application

See also

Property: `Application` as `Application` (read-only)

Description
Access the XMLSpy application object.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

DiagramFormat

See also

Property: `DiagramFormat` as SPYImageKind

Description
Set this property to true , to specify the diagram image type. This property is not available for HTML documentation. The property is initialized with the value used during the last call to `DocumentGenerateWSDLDocumentation`. The default for the first run is PNG.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

EmbedDiagrams

See also

Property: EmbedDiagrams as Boolean

Description
Set this property to true, to embed the diagrams in the generated document. This property is not available for HTML documentation. The property is initialized with the value used during the last call to DocumentGenerateWSDLDocumentation. The default for the first run is true.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

IncludeAll

See also

Method: IncludeAll (i_bInclude as Boolean)

Description
Use this method to mark or unmark all include options.

Errors
3900 The object is no longer valid.

IncludeBinding

See also

Property: IncludeBinding as Boolean

Description
Set this property to true, to include bindings in the WSDL documentation. The property is initialized with the value used during the last call to DocumentGenerateWSDLDocumentation. The default for the first run is true.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

IncludeImportedWSDLFiles

See also

Property: IncludeImportedWSDLFiles as Boolean

Description
Set this property to true, to include imported WSDL finles in the WSDL documentation. The property is initialized with the value used during the last call to DocumentGenerateWSDLDocumentation. The default for the first run is true.

 © 2007 Altova GmbH

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

IncludeMessages

See also

Property: IncludeMessages as Boolean

Description
Set this property to true, to include messages in the WSDL documentation. The property is initialized with the value used during the last call to DocumentGenerateWSDLDocumentation. The default for the first run is true.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

IncludeOverview

See also

Property: IncludeOverview as Boolean

Description
Set this property to true, to include an overview in the WSDL documentation. The property is initialized with the value used during the last call to DocumentGenerateWSDLDocumentation. The default for the first run is true.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

IncludePortType

See also

Property: IncludePortType as Boolean

Description
Set this property to true, to include port types in the WSDL documentation. The property is initialized with the value used during the last call to DocumentGenerateWSDLDocumentation. The default for the first run is true.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

IncludeService

See also

Property: IncludeService as Boolean

Description
Set this property to true, to include services in the WSDL documentation. The property is initialized with the value used during the last call to DocumentGenerateWSDLDocumentation. The default for the first run is true.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

IncludeTypes

See also

Property: `IncludeTypes` as Boolean

Description
Set this property to true, to include types in the WSDL documentation. The property is initialized with the value used during the last call to DocumentGenerateWSDLDocumentation. The default for the first run is true.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

MultipleOutputFiles

See also

Property: `MultipleOutputFiles` as Boolean

Description
Set this property to true, to split the documentation files. The property is initialized with the value used during the last call to DocumentGenerateWSDLDocumentation. The default for the first run is false.

Errors
3900 The object is no longer valid.
3901 Invalid value has been used to set the property.
Invalid address for the return parameter was specified.

OptionsDialogAction

See also

Property: `OptionsDialogAction` as SPYDialogAction

Description
To allow your script to fill in the default values and let the user see and react on the dialog, set this property to the value *spyDialogUserInput(2)*. If you want your script to define all the options in the schema documentation dialog without any user interaction necessary, use *spyDialogOK(0)*. Default is *spyDialogOK.*

Errors
3900 The object is no longer valid.

© 2007 Altova GmbH

3901 Invalid value has been used to set the property.
Invalid address for the return parameter was specified.

OutputFile

See also

Property: `OutputFile` as String

Description
Full path and name of the file that will contain the generated documentation. In case of HTML output, additional '.png' files will be generated based on this filename. The default value for this property is an empty string and needs to be replaced before using this object in a call to DocumentGenerateWSDLDocumentation.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

OutputFileDialogAction

See also

Property: `OutputFileDialogAction` as SPYDialogAction

Description
To allow the user to select the output file with a file selection dialog, set this property to *spyDialogUserInput(2)*. If the value stored in OutputFile should be taken and no user interaction should occur, use *spyDialogOK(0)*. Default is *spyDialogOK.*

Errors
3900 The object is no longer valid.
3901 Invalid value has been used to set the property.
Invalid address for the return parameter was specified.

OutputFormat

See also

Property: `OutputFormat` as SPYSchemaDocumentationFormat

Description
Defines the kind of documentation that will be generated: HTML (value=0), MS-Word (value=1), or RTF (value=2). The property gets initialized with the value used during the last call to DocumentGenerateWSDLDocumentation. The default for the first run is HTML.

Errors
3900 The object is no longer valid.
3901 Invalid value has been used to set the property.
Invalid address for the return parameter was specified.

Parent

See also

Property: Parent as Dialogs (read-only)

Description
Access the parent of the object.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

ShowBindingDiagram

See also

Property: ShowBindingDiagram as Boolean

Description
Set this property to true, to show binding diagrams in the WSDL documentation. The property is initialized with the value used during the last call to DocumentGenerateWSDLDocumentation. The default for the first run is true.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

ShowExtensibility

See also

Property: ShowExtensibility as Boolean

Description
Set this property to true, to show service and binding extensibilities in the WSDL documentation. The property is initialized with the value used during the last call to DocumentGenerateWSDLDocumentation. The default for the first run is true.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

ShowMessageParts

See also

Property: ShowMessageParts as Boolean

Description
Set this property to true, to show message parts of messges in the WSDL documentation. The property is initialized with the value used during the last call to DocumentGenerateWSDLDocumentation. The default for the first run is true.

© 2007 Altova GmbH

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

ShowPort

See also

Property: ShowPort as Boolean

Description
Set this property to true, to show service ports in the WSDL documentation. The property is initialized with the value used during the last call to DocumentGenerateWSDLDocumentation. The default for the first run is true.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

ShowPortTypeDiagram

See also

Property: ShowPortTypeDiagram as Boolean

Description
Set this property to true, to show port type diagrams in the WSDL documentation. The property is initialized with the value used during the last call to DocumentGenerateWSDLDocumentation. The default for the first run is true.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

ShowPortTypeOperations

See also

Property: ShowPortTypeOperations as Boolean

Description
Set this property to true, to show port type operations in the WSDL documentation. The property is initialized with the value used during the last call to DocumentGenerateWSDLDocumentation. The default for the first run is true.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

ShowProgressBar

See also

Property: ShowProgressBar as Boolean

Description
Set this property to true, to make the window showing the document generation progress visible. Use false, to hide it. Default is false.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

ShowResult

See also

Property: ShowResult as Boolean

Description
Set this property to true, to automatically open the resulting document when generation was successful. HTML documentation will be opened in XMLSpy. To show Word documentation, MS-Word will be started. The property gets initialized with the value used during the last call to DocumentGenerateWSDLDocumentation. The default for the first run is true.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

ShowServiceDiagram

See also

Property: ShowServiceDiagram as Boolean

Description
Set this property to true, to show service diagrams in the WSDL documentation. The property is initialized with the value used during the last call to DocumentGenerateWSDLDocumentation. The default for the first run is true.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

ShowSourceCode

See also

Property: ShowSourceCode as Boolean

Description
Set this property to true, to show source code for the includes in the WSDL documentation. The property is initialized with the value used during the last call to

 © 2007 Altova GmbH

DocumentGenerateWSDLDocumentation. The default for the first run is true.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

ShowTypesDiagram

See also

Property: ShowTypesDiagram as Boolean

Description
Set this property to true, to show type diagrams in the WSDL documentation. The property is initialized with the value used during the last call to DocumentGenerateWSDLDocumentation. The default for the first run is true.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

ShowUsedBy

See also

Property: ShowUsedBy as Boolean

Description
Set this property to true, to show the used-by relation for types, bindings and messages definitions in the WSDL documentation. The property is initialized with the value used during the last call to DocumentGenerateWSDLDocumentation. The default for the first run is true.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

4.3.28 XMLData

See also

Properties
Kind
Name
TextValue

HasChildren
MayHaveChildren
Parent

Methods
GetFirstChild
GetNextChild
GetCurrentChild

InsertChild
AppendChild

EraseAllChildren
EraseCurrentChild

IsSameNode

CountChildren
CountChildrenKind

GetChild
GetChildKind

HasChildrenKind

Description
The XMLData interface provides direct XML-level access to a document. You can read and directly modify the XML representation of the document. However, please, note the following restrictions:

- The XMLData representation is only valid when the document is shown in grid view or authentic view.
- When in authentic view, additional XMLData elements are automatically inserted as parents of each visible document element. Typically this is an XMLData of kind spyXMLDataElement with the Name property set to 'Text'.
- When you use the XMLData interface while in a different view mode you will not receive errors, but changes are not reflected to the view and might get lost during the next view switch.

Objects of this class represent the different atomic parts of an XML document. See the enumeration type SPYXMLDataKind for the available part types. Each part knows its children, thus forming a XMLData tree with DocumentRootElement at its top. To get the top element of the document content - ignoring the XML header - use DocumentDataRoot. For examples on how to traverse the XMLData tree see UsingXMLDatatomodifydocumentstructure and GetNextChild.

AppendChild

See also

Declaration: AppendChild (*pNewData* as XMLData)

Description
AppendChild appends pNewData as last child to the XMLData object. See also Using XMLData.

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
1900	Document must not be modified

Example

```
Dim objCurrentParent As XMLData
```


© 2007 Altova GmbH

```
Dim objNewChild As XMLData

Set objNewChild = objSpy.ActiveDocument.CreateChild(spyXMLDataElement)
Set objCurrentParent = objSpy.ActiveDocument.RootElement

objCurrentParent.AppendChild objNewChild

Set objNewChild = Nothing
```

CountChildren

See also

Declaration: `CountChildren` as long

Description
`CountChildren` gets the number of children.

Available with TypeLibrary version 1.5

Errors
1500 The XMLData object is no longer valid.

CountChildrenKind

See also

Declaration: `CountChildrenKind` (*nKind* as SPYXMLDataKind) as long

Description
`CountChildrenKind` gets the number of children of the specific kind.

Available with TypeLibrary version 1.5

Errors
1500 The XMLData object is no longer valid.

EraseAllChildren

See also

Declaration: `EraseAllChildren`

Description
`EraseAllChildren` deletes all associated children of the `XMLData` object.

Errors
1500 The XMLData object is no longer valid.
1900 Document must not be modified

Example
The sample erases all elements of the active document.

```
Dim objCurrentParent As XMLData

Set objCurrentParent = objSpy.ActiveDocument.RootElement
objCurrentParent.EraseAllChildren
```

EraseCurrentChild

See also

Declaration: `EraseCurrentChild`

Description

`EraseCurrentChild` deletes the current `XMLData` child object. Before you call `EraseCurrentChild` you must initialize an internal iterator with `XMLData.GetFirstChild`. After deleting the current child, `EraseCurrentChild` increments the internal iterator of the XMLData element. No error is returned when the last child gets erased and the iterator is moved past the end of the child list. The next call to `EraseCurrentChild` however, will return error 1503.

Errors

1500 The XMLData object is no longer valid.
1503 No iterator is initialized for this XMLData object, or the iterator points past the last child.
1900 Document must not be modified

Examples

```
// ---------------------------------------
// XMLSpy scripting environment - JScript
// erase all children of XMLData
// ---------------------------------------
// let's get an XMLData element, we assume that the
// cursor selects the parent of a list in grid view
var objList = Application.ActiveDocument.GridView.CurrentFocus;

// the following line would be shorter, of course
//objList.EraseAllChildren ();

// but we want to demonstrate the usage of EraseCurrentChild
if ((objList != null) && (objList.HasChildren))
{
 try
 {
  objEle = objList.GetFirstChild(-1);
  while (objEle != null)
   objList.EraseCurrentChild();
   // no need to call GetNextChild
 }
 catch (err)
  // 1503 - we reached end of child list
  { if ((err.number & 0xffff) != 1503) throw (err); }
}
```

GetChild

See also

Declaration: `GetChild` (*`position`* as long) as `XMLData`

Return Value

Returns an XML element as `XMLData` object.

© 2007 Altova GmbH

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.

GetChildKind

See also

Declaration: GetChildKind (*position* as long, *nKind* as SPYXMLDataKind) as XMLData

Return Value
Returns an XML element as XMLData object.

Description
GetChildKind() returns a reference to a child of this kind at the given index (zero-based). The position parameter is relative to the number of children of the specified kind and not to all children of the object.

Available with TypeLibrary version 1.5

Errors
1500 The XMLData object is no longer valid.
1510 Invalid address for the return parameter was specified.

GetCurrentChild

See also

Declaration: GetCurrentChild as XMLData

Return Value
Returns an XML element as XMLData object.

Description
GetCurrentChild gets the current child. Before you call GetCurrentChild you must initialize an internal iterator with XMLData.GetFirstChild.

Errors
1500 The XMLData object is no longer valid.
1503 No iterator is initialized for this XMLData object.
1510 Invalid address for the return parameter was specified.

GetFirstChild

See also

Declaration: GetFirstChild (*nKind* as SPYXMLDataKind) as XMLData

Return Value
Returns an XML element as XMLData object.

Description
GetFirstChild initializes a new iterator and returns the first child. Set nKind = -1 to get an iterator for all kinds of children.
REMARK: The iterator is stored inside the XMLData object and gets destroyed when the XMLData object gets destroyed. Be sure to keep a reference to this object as long as you want to use GetCurrentChild, GetNextChild or EraseCurrentChild.

Errors
1500 The XMLData object is no longer valid.
1501 Invalid XMLData kind was specified.
1504 Element has no children of specified kind.
1510 Invalid address for the return parameter was specified.

Example
See the example at XMLData.GetNextChild.

GetNextChild

See also

Declaration: GetNextChild as XMLData

Return Value
Returns an XML element as XMLData object.

Description
GetNextChild steps to the next child of this element. Before you call GetNextChild you must initialize an internal iterator with XMLData.GetFirstChild.

Check for the last child of the element as shown in the sample below.

Errors
1500 The XMLData object is no longer valid.
1503 No iterator is initialized for this XMLData object.
1510 Invalid address for the return parameter was specified.

Examples

```
' ------------------------------------------------
' VBA code snippet - iterate XMLData children
' ------------------------------------------------
On Error Resume Next
Set objParent = 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)
```

 © 2007 Altova GmbH

```
// ---------------------------------------
// 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);
}
```

HasChildrenKind

See also

Declaration: `HasChildrenKind` (*nKind* as `SPYXMLDataKind`) as Boolean

Description
The method returns true if the object is the parent of other `XMLData` objects of the specific kind.

Available with TypeLibrary version 1.5

Errors
1500 The XMLData object is no longer valid.
1510 Invalid address for the return parameter was specified.

HasChildren

See also

Declaration: `HasChildren` as Boolean

Description
The property is true if the object is the parent of other `XMLData` objects. This property is read-only.

Errors
1500 The XMLData object is no longer valid.
1510 Invalid address for the return parameter was specified.

InsertChild

See also

Declaration: `InsertChild` (*pNewData* as `XMLData`)

Description
`InsertChild` inserts the new child before the current child (see also `XMLData.GetFirstChild`, `XMLData.GetNextChild` to set the current child).

Errors
1500 The XMLData object is no longer valid.
1503 No iterator is initialized for this XMLData object.
1505 Invalid XMLData kind was specified.
1506 Invalid address for the return parameter was specified.
1507 Element cannot have Children
1512 Cyclic insertion - new data element is already part of document
1900 Document must not be modified

Examples
See Using XMLData to modify document structure.

IsSameNode

See also

Declaration: `IsSameNode` (*pNodeToCompare* as `XMLData`) as Boolean

Description

Returns true if `pNodeToCompare` references the same node as the object itself.

Errors
1500 The XMLData object is no longer valid.
1506 Invalid address for the return parameter was specified.

Kind

See also

Declaration: `Kind` as `SPYXMLDataKind`

Description
Kind of this `XMLData` object. This property is read-only.

Errors
1500 The XMLData object is no longer valid.
1510 Invalid address for the return parameter was specified.

MayHaveChildren

See also

Declaration: `MayHaveChildren` as Boolean

© 2007 Altova GmbH

Description
Indicates whether it is allowed to add children to this XMLData object.
This property is read-only.

Errors
1500 The XMLData object is no longer valid.
1510 Invalid address for the return parameter was specified.

Name

See also

Declaration: Name as String

Description
Used to modify and to get the name of the XMLData object.

Errors
1500 The XMLData object is no longer valid.
1510 Invalid address for the return parameter was specified.

Parent

See also

Declaration: Parent as XMLData

Return value
Parent as XMLData object. Nothing (or NULL) if there is no parent element.

Description
Parent of this element. This property is read-only.

Errors
1500 The XMLData object is no longer valid.
1510 Invalid address for the return parameter was specified.

TextValue

See also

Declaration: TextValue as String

Description
Used to modify and to get the text value of this XMLData object.

Errors
1500 The XMLData object is no longer valid.
1510 Invalid address for the return parameter was specified.

Examples
See Using XMLData to modify document structure.

4.4 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.

4.4.1 AuthenticEvent (obsolete)

Superseded by AuthenticView and AuthenticRange

The DocEditView object is renamed to OldAuthenticView.
DocEditSelection is renamed to AuthenticSelection.
DocEditEvent is renamed to AuthenticEvent.
DocEditDataTransfer is renamed to AuthenticDataTransfer.

Their usage - except for AuthenticDataTransfer - is no longer recommended. We will continue to support existing functionality for a yet undefined period of time but no new features will be added to these interface. All functionality available up to now in DocEditView, DocEditSelection, DocEditEvent and DocEditDataTransfer is now available via AuthenticView, AuthenticRange and AuthenticDataTransfer. Many new features have been added.

For examples on migrating from DocEdit to Authentic see the description of the different methods and properties of the different DocEdit objects.

See also

Properties

altKey
altLeft
ctrlKey
ctrlLeft
shiftKey
shiftLeft

keyCode
repeat

button

clientX
clientY

dataTransfer

srcElement
fromElement

propertyName

cancelBubble
returnValue

type

Description

DocEditEvent interface.

© 2007 Altova GmbH

altKey (obsolete)

Superseded by parameters to
AuthenticView.OnKeyboardEvent (On_AuthenticView_KeyPressed)
AuthenticView.OnMouseEvent (On_AuthenticView_MouseEvent)
AuthenticView.OnDragOver (On_AuthenticView_DragOver)

The event object that holds the information of the last event is now replaced by parameters to the different event handler functions to simplify data access. The event object will be supported for a not yet defined period of time for compatibility reasons. No improvements are planned. It is highly recommended to migrate to the new event handler functions.

```
// ----- XMLSpy scripting environment - javascript sample -----
// instead of:
// function On_DocEditKeyPressed ()
// {
//     if (Application.ActiveDocument.DocEditView.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");
}
```

See also

Declaration: `altKey` as Boolean

Description
True if the right ALT key is pressed.

altLeft (obsolete)

Superseded by parameters to
AuthenticView.OnKeyboardEvent (On_AuthenticView_KeyPressed)
AuthenticView.OnMouseEvent (On_AuthenticView_MouseEvent)
AuthenticView.OnDragOver (On_AuthenticView_DragOver)

The event object that holds the information of the last event is now replaced by parameters to the different event handler functions to simplify data access. The event object will be supported for a not yet defined period of time for compatibility reasons. No improvements are planned. It is highly recommended to migrate to the new event handler functions.

```
// ----- XMLSpy scripting environment - javascript sample -----
// instead of:
// function On_DocEditKeyDown ()
// {
//     if (Application.ActiveDocument.DocEditView.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");
}
```

See also

Declaration: `altLeft` as Boolean

Description
True if the left ALT key is pressed.

© 2007 Altova GmbH

button (obsolete)

Superseded by parameters to
AuthenticView.OnMouseEvent (On_AuthenticView_MouseEvent)
AuthenticView.OnDragOver (On_AuthenticView_DragOver)

The event object that holds the information of the last event is now replaced by parameters to the different event handler functions to simplify data access. The event object will be supported for a not yet defined period of time for compatibility reasons. No improvements are planned. It is highly recommended to migrate to the new event handler functions.

```
// ----- XMLSpy scripting environment - javascript sample -----
// instead of:
// function On_DocEditButtonDown ()
// {
//     if ( Application. ActiveDocument. DocEditView. 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");
}
```

See also

Declaration: `button` as long

Description
Specifies which mouse button is pressed:

0	No button is pressed.
1	Left button is pressed.
2	Right button is pressed.
3	Left and right buttons are both pressed.
4	Middle button is pressed.
5	Left and middle buttons both are pressed.
6	Right and middle buttons are both pressed.
7	All three buttons are pressed.

cancelBubble (obsolete)

Superseded by the boolean return value of following event handler functions
AuthenticView.OnKeyboardEvent (On_AuthenticView_KeyPressed)
AuthenticView.OnMouseEvent (On_AuthenticView_MouseEvent)
AuthenticView.OnDragOver (On_AuthenticView_DragOver)

The event object that holds the information of the last event is now replaced by parameters to the different event handler functions to simplify data access. The event object will be supported for a not yet defined period of time for compatibility reasons. No improvements are planned. It is highly recommended to migrate to the new event handler functions.

Returning *true* from an event handler function signals that the event has beend handled and normal event handling should be aborted.

```
// ----- XMLSpy scripting environment - javascript sample -----
// instead of:
// function On_DocEditKeyPressed ()
// {
//     if (Application.ActiveDocument.DocEditView.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 :-)
}
```

See also

Declaration: `cancelBubble` as Boolean

Description
Set `cancelBubble` to TRUE if the default event handler should not be called.

© 2007 Altova GmbH

clientX (obsolete)

Superseded by parameters to
AuthenticView.OnMouseEvent (On_AuthenticView_MouseEvent)
AuthenticView.OnBeforeDrop (On_AuthenticView_BeforeDrop)
AuthenticView.OnDragOver (On_AuthenticView_DragOver)

The event object that holds the information of the last event is now replaced by parameters to the different event handler functions to simplify data access. The event object will be supported for a not yet defined period of time for compatibility reasons. No improvements are planned. It is highly recommended to migrate to the new event handler functions.

```
// ----- XMLSpy scripting environment - javascript sample -----
// instead of:
// function On_DocEditMouseMove ()
// {
//     MsgBox ("moving over " +
Application.ActiveDocument.DocEditView.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);
}
```

See also

Declaration: `clientX` as long

Description
X value of the current mouse position in client coordinates.

clientY (obsolete)

Superseded by parameters to
AuthenticView.OnMouseEvent (On_AuthenticView_MouseEvent)
AuthenticView.OnBeforeDrop (On_AuthenticView_BeforeDrop)
AuthenticView.OnDragOver (On_AuthenticView_DragOver)

The event object that holds the information of the last event is now replaced by parameters to the different event handler functions to simplify data access. The event object will be supported for a not yet defined period of time for compatibility reasons. No improvements are planned. It is highly recommended to migrate to the new event handler functions.

```
// ----- XMLSpy scripting environment - javascript sample -----
// instead of:
// function On_DocEditMouseMove ()
// {
//     MsgBox ("moving over " +
Application.ActiveDocument.DocEditView.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);
}
```

See also

Declaration: `clientY` as long

Description
Y value of the current mouse position in client coordinates.

© 2007 Altova GmbH

ctrlKey (obsolete)

Superseded by parameters to
AuthenticView.OnKeyboardEvent (On_AuthenticView_KeyPressed)
AuthenticView.OnMouseEvent (On_AuthenticView_MouseEvent)
AuthenticView.OnDragOver (On_AuthenticView_DragOver)

The event object that holds the information of the last event is now replaced by parameters to the different event handler functions to simplify data access. The event object will be supported for a not yet defined period of time for compatibility reasons. No improvements are planned. It is highly recommended to migrate to the new event handler functions.

```
// ----- XMLSpy scripting environment - javascript sample -----
// instead of:
// function On_DocEditMouseMove ()
// {
//     if (Application.ActiveDocument.DocEditView.event.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");
}
```

See also

Declaration: `ctrlKey` as Boolean

Description
True if the right CTRL key is pressed.

ctrlLeft (obsolete)

Superseded by parameters to
AuthenticView.OnKeyboardEvent (On_AuthenticView_KeyPressed)
AuthenticView.OnMouseEvent (On_AuthenticView_MouseEvent)
AuthenticView.OnDragOver (On_AuthenticView_DragOver)

The event object that holds the information of the last event is now replaced by parameters to the different event handler functions to simplify data access. The event object will be supported for a not yet defined period of time for compatibility reasons. No improvements are planned. It is highly recommended to migrate to the new event handler functions.

```
// ----- XMLSpy scripting environment - javascript sample -----
// instead of:
// function On_DocEditMouseMove ()
// {
//     if (Application.ActiveDocument.DocEditView.event.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");
}
```

See also

Declaration: `ctrlLeft` as Boolean

Description
True if the left CTRL key is pressed.

© 2007 Altova GmbH

dataTransfer (obsolete)

Superseded by parameters to
AuthenticView.OnBeforeDrop (On_AuthenticView_BeforeDrop)
AuthenticView.OnDragOver (On_AuthenticView_DragOver)

The event object that holds the information of the last event is now replaced by parameters to the different event handler functions to simplify data access. The event object will be supported for a not yet defined period of time for compatibility reasons. No improvements are planned. It is highly recommended to migrate to the new event handler functions.

```
// ----- XMLSpy scripting environment - javascript sample -----
// instead of:
// function On_DocEditDrop ()
// {
//     if ( Application. ActiveDocument. DocEditView. 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;
}
```

See also

Declaration: `dataTransfer` as Variant

Description
Property dataTransfer

fromElement (obsolete)

Not supported

See also

Declaration: `fromElement` as Variant (not supported)

Description
Currently no event sets this property.

keyCode (obsolete)

Superseded by a parameter to AuthenticView.OnKeyboardEvent (On_AuthenticView_KeyPressed)

The event object that holds the information of the last event is now replaced by parameters to the different event handler functions to simplify data access. The event object will be supported for a not yet defined period of time for compatibility reasons. No improvements are planned. It is highly recommended to migrate to the new event handler functions.

```
// ----- XMLSpy scripting environment - javascript sample -----
// instead of:
// function On_DocEditKeyPressed ()
// {
//     if (Application.ActiveDocument.DocEditView.event.keyCode == 0x20)
//     {
//        // cancel key processing, swallow spaces :-)
//        Application.ActiveDocument.DocEditView.event.cancelBubble = true;
//     }
// }
// use now:
function On_AuthenticView_KeyPressed (SPYKeyEvent i_eKeyEvent, long
i_nKeyCode, SPYVirtualKeyMask i_nVirtualKeyStatus)
{
    if (i_nKeyCode == 0x20)
        return true;  // cancel key processing, swallow spaces :-)
}
```

See also

Declaration: `keyCode` as long

Description
Keycode of the currently pressed key. This property is read-write.

© 2007 Altova GmbH

propertyName (obsolete)

Not supported

See also

Declaration: `propertyName` as String (not supported)

Description
Currently no event sets this property.

repeat (obsolete)

Not supported

See also

Declaration: `repeat` as Boolean (not supported)

Description
True if the `onkeydown` event is repeated.

returnValue (obsolete)

No longer supported

See also

Declaration: `returnValue` as Variant

Description
Use `returnValue` to set a return value for your event handler.

shiftKey (obsolete)

Superseded by parameters to
AuthenticView.OnKeyboardEvent (On_AuthenticView_KeyPressed)
AuthenticView.OnMouseEvent (On_AuthenticView_MouseEvent)
AuthenticView.OnDragOver (On_AuthenticView_DragOver)

The event object that holds the information of the last event is now replaced by parameters to the different event handler functions to simplify data access. The event object will be supported for a not yet defined period of time for compatibility reasons. No improvements are planned. It is highly recommended to migrate to the new event handler functions.

```
// ----- XMLSpy scripting environment - javascript sample -----
// instead of:
// function On_DocEditDragOver ()
// {
//     if (Application.ActiveDocument.DocEditView.event.shiftKey ||
//         Application.ActiveDocument.DocEditView.event.shiftLeft)
//       MsgBox ("shift key is down");
// }
// use now:
function On_AuthenticView_DragOver (long i_nXPos, long i_nYPos,
                                    SPYMouseEvent i_eMouseEvent,
                                    IAuthenticRange *i_ipRange,
                                    IAuthenticDataTransfer *i_ipData)
{
    if (i_eMouseEvent & spyShiftKeyMask)
        MsgBox ("shift key is down");
}
```

See also

Declaration: `shiftKey` as Boolean

Description
True if the right SHIFT key is pressed.

© 2007 Altova GmbH

shiftLeft (obsolete)

Superseded by parameters to
AuthenticView.OnKeyboardEvent (On_AuthenticView_KeyPressed)
AuthenticView.OnMouseEvent (On_AuthenticView_MouseEvent)
AuthenticView.OnDragOver (On_AuthenticView_DragOver)

The event object that holds the information of the last event is now replaced by parameters to the different event handler functions to simplify data access. The event object will be supported for a not yet defined period of time for compatibility reasons. No improvements are planned. It is highly recommended to migrate to the new event handler functions.

```
// ----- XMLSpy scripting environment - javascript sample -----
// instead of:
// function On_DocEditDragOver ()
// {
//     if (Application.ActiveDocument.DocEditView.event.shiftKey ||
//         Application.ActiveDocument.DocEditView.event.shiftLeft)
//       MsgBox ("shift key is down");
// }
// use now:
function On_AuthenticView_DragOver (long i_nXPos, long i_nYPos,
                                    SPYMouseEvent i_eMouseEvent,
                                    IAuthenticRange *i_ipRange,
                                    IAuthenticDataTransfer *i_ipData)
{
    if (i_eMouseEvent & spyShiftKeyMask)
        MsgBox ("shift key is down");
}
```

See also

Declaration: `shiftLeft` as Boolean

Description
True if the left SHIFT key is pressed.

srcElement (obsolete)

Superseded by parameters to
AuthenticView.OnMouseEvent (On_AuthenticView_MouseEvent)
AuthenticView.OnBeforeDrop (On_AuthenticView_BeforeDrop)
AuthenticView.OnDragOver (On_AuthenticView_DragOver)

The event object that holds the information of the last event is now replaced by parameters to the different event handler functions to simplify data access. The event object will be supported for a not yet defined period of time for compatibility reasons. No improvements are planned. It is highly recommended to migrate to the new event handler functions.

With the new event handler function, a range object selecting this element is provided instead of the XMLData element currently below the mouse cursor.

```
// ----- XMLSpy scripting environment - javascript sample -----
// instead of:
// function On_DocEditMouseMove ()
// {
//     var objEvent = Application.ActiveDocument.DocEditView.event;
//     if (objEvent.srcElement != null)
//         MsgBox ("moving over " + objEvent.srcElement.Parent.Name);
// }
// 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);
}
```

See also

Declaration: `srcElement` as Variant

Description
Element which fires the current event. This is usually an `XMLData` object.

© 2007 Altova GmbH

type (obsolete)

Not supported

See also

Declaration: `type` as String (not supported)

Description
Currently no event sets this property.

4.4.2 AuthenticSelection (obsolete)

Superseded by AuthenticRange

The DocEditView object is renamed to OldAuthenticView.
DocEditSelection is renamed to AuthenticSelection.
DocEditEvent is renamed to AuthenticEvent.
DocEditDataTransfer is renamed to AuthenticDataTransfer.

Their usage - except for AuthenticDataTransfer - is no longer recommended. We will continue to support existing functionality for a yet undefined period of time but no new features will be added to these interface. All functionality available up to now in DocEditView, DocEditSelection, DocEditEvent and DocEditDataTransfer is now available via AuthenticView, AuthenticRange and AuthenticDataTransfer. Many new features have been added.

For examples on migrating from DocEdit to Authentic see the description of the different methods and properties of the different DocEdit objects.

See also

Properties
Start
StartTextPosition
End
EndTextPosition

© 2007 Altova GmbH

End (obsolete)

Superseded by AuthenticRange.LastXMLData

```
// ----- javascript sample -----
// instead of:
// var objXMLData =
Application.ActiveDocument.DocEditView.CurrentSelection.End;
// use now:
var objXMLData =
Application.ActiveDocument.AuthenticView.Selection.LastXMLData;
```

See also

Declaration: `End` as `XMLData`

Description
XML element where the current selection ends.

EndTextPosition (obsolete)

Superseded by AuthenticRange.LastXMLDataOffset

```
// ----- javascript sample -----
// instead of:
// var nOffset =
Application.ActiveDocument.DocEditView.CurrentSelection.EndTextPosition;
// use now:
var nOffset =
Application.ActiveDocument.AuthenticView.Selection.LastXMLDataOffset;
```

See also

Declaration: `EndTextPosition` as long

Description
Position in `DocEditSelection.End.TextValue` where the selection ends.

Start (obsolete)

Superseded by AuthenticRange.FirstXMLData

```
// ----- javascript sample -----
// instead of:
// var objXMLData =
Application.ActiveDocument.DocEditView.CurrentSelection.Start;
// use now:
var objXMLData =
Application.ActiveDocument.AuthenticView.Selection.FirstXMLData;
```

See also

Declaration: `Start` as `XMLData`

Description
XML element where the current selection starts.

StartTextPosition (obsolete)

Superseded by AuthenticRange.FirstXMLDataOffset

```
// ----- javascript sample -----
// instead of:
// var nOffset =
Application.ActiveDocument.DocEditView.CurrentSelection.StartTextPosition;
// use now:
var nOffset =
Application.ActiveDocument.AuthenticView.Selection.FirstXMLDataOffset;
```

See also

Declaration: `StartTextPosition` as long

Description
Position in `DocEditSelection.Start.TextValue` where the selection starts.

© 2007 Altova GmbH

4.4.3 OldAuthenticView (obsolete)

Superseded by AuthenticView and AuthenticRange

The DocEditView object is renamed to OldAuthenticView.
DocEditSelection is renamed to AuthenticSelection.
DocEditEvent is renamed to AuthenticEvent.
DocEditDataTransfer is renamed to AuthenticDataTransfer.

Their usage - except for AuthenticDataTransfer - is no longer recommended. We will continue to support existing functionality for a yet undefined period of time but no new features will be added to these interfaces. All functionality available up to now in DocEditView, DocEditSelection, DocEditEvent and DocEditDataTransfer is now available via AuthenticView, AuthenticRange and AuthenticDataTransfer. Many new features have been added.

For examples on migrating from DocEdit to Authentic see the description of the different methods and properties of the different DocEdit objects.

See also

Methods

LoadXML
SaveXML

EditClear
EditCopy
EditCut
EditPaste
EditRedo
EditSelectAll
EditUndo

RowAppend
RowDelete
RowDuplicate
RowInsert
RowMoveDown
RowMoveUp

ApplyTextState
IsTextStateApplied
IsTextStateEnabled

MarkUpView

SelectionSet
SelectionMoveTabOrder

GetNextVisible
GetPreviousVisible

GetAllowedElements

Properties
CurrentSelection

event

XMLRoot

IsEditClearEnabled
IsEditCopyEnabled
IsEditCutEnabled
IsEditPasteEnabled
IsEditRedoEnabled
IsEditUndoEnabled

IsRowAppendEnabled
IsRowDeleteEnabled
IsRowDuplicateEnabled
IsRowInsertEnabled
IsRowMoveDownEnabled
IsRowMoveUpEnabled

Description
Interface for Authentic.

© 2007 Altova GmbH

ApplyTextState (obsolete)

Superseded by AuthenticRange.PerformAction

Use spyAuthenticApply for the eAction parameter. The PerformAction method allows to apply text state attributes to any range of the document, not only the current UI selection.

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.DocEditView.ApplyTextState ("bold");
// use now:
if (! Application.ActiveDocument.AuthenticView.Selection.PerformAction
(spyAuthenticApply, "bold"))
    MsgBox ("Error: can't set current selection to bold");
```

See also

Declaration: `ApplyTextState` (*`elementName`* as String)

Description
Applies or removes the text state defined by the parameter `elementName`. Common examples for the parameter elementName would be strong and italic.

In an XML document there are segments of data, which may contain sub-elements. For example consider the following HTML:

```
<b>fragment</b>
```

The HTML tag `<b>` will cause the word fragment to be bold. However, this only happens because the HTML parser knows that the tag `<b>` is bold. With XML there is much more flexibility. It is possible to define any XML tag to do anything you desire. The point is that it is possible to apply a Text state using XML. But the Text state that is applied must be part of the schema. For example in the `OrgChart.xml, OrgChart.sps, OrgChart.xsd` example the tag `<strong>` is the same as bold. And to apply bold the method `ApplyTextState()` is called. But like the row and edit operations it is necessary to test if it is possible to apply the text state.

See also `IsTextStateEnabled` and `IsTextStateApplied`.

CurrentSelection (obsolete)

Superseded by AuthenticView.Selection

The returned AuthenticRange object supports navigation via XMLData elements as well as navigation by document elements (e.g. characters, words, tags) or text cursor positions.

```
// ----- javascript sample -----
// instead of:
// var objDocEditSel =
Application.ActiveDocument.DocEditView.CurrentSelection;
// use now:
    var objRange = Application.ActiveDocument.AuthenticView.Selection;
```

See also

Declaration: `CurrentSelection` as `DocEditSelection`

Description
The property provides access to the current selection in the Authentic.

EditClear (obsolete)

Superseded by AuthenticRange.Delete

The Delete method of AuthenticRange allows to delete any range of the document, not only the current UI selection.

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.DocEditView.EditClear();
// use now:
if (! Application.ActiveDocument.AuthenticView.Selection.Delete())
    MsgBox ("Error: can't delete current selection");
```

See also

Declaration: `EditClear`

Description
Deletes the current selection.

© 2007 Altova GmbH

EditCopy (obsolete)

Superseded by AuthenticRange.Copy

The Copy method of AuthenticRange allows to delete any range of the document, not only the current UI selection.

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.DocEditView.EditCopy();
// use now:
if (! Application.ActiveDocument.AuthenticView.Selection.Copy())
    MsgBox ("Error: can't copy current selection");
```

See also

Declaration: `EditCopy`

Description
Copies the current selection to the clipboard.

EditCut (obsolete)

Superseded by AuthenticRange.Cut

The Cut method of AuthenticRange allows to delete any range of the document, not only the current UI selection.

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.DocEditView.EditCut();
// use now:
if (! Application.ActiveDocument.AuthenticView.Selection.Cut())
    MsgBox ("Error: can't cut out current selection");
```

See also

Declaration: `EditCut`

Description
Cuts the current selection from the document and copies it to the clipboard.

EditPaste (obsolete)

Superseded by AuthenticRange.Paste

The Paste method of AuthenticRange allows to delete any range of the document, not only the current UI selection.

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.DocEditView.EditPaste();
// use now:
if (! Application.ActiveDocument.AuthenticView.Selection.Paste())
    MsgBox ("Error: can't paste to current selection");
```

See also

Declaration: `EditPaste`

Description
Pastes the content from the clipboard into the document.

EditRedo (obsolete)

Superseded by AuthenticView.Redo

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.DocEditView.EditRedo();
// use now:
if (! Application.ActiveDocument.AuthenticView.Redo())
    MsgBox ("Error: no redo step available");
```

See also

Declaration: `EditRedo`

Description
Redo the last undo step.

© 2007 Altova GmbH

EditSelectAll (obsolete)

Superseded by AuthenticView.WholeDocument and AuthenticRange.Select

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.DocEditView.EditSelectAll();
// use now:
Application.ActiveDocument.AuthenticView.WholeDocument.Select();
```

See also

Declaration: `EditSelectAll`

Description
The method selects the complete document.

EditUndo (obsolete)

Superseded by AuthenticView.Undo

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.DocEditView.EditUndo();
// use now:
if (! Application.ActiveDocument.AuthenticView.Undo())
    MsgBox ("Error: no undo step available");
```

See also

Declaration: `EditUndo`

Description
Undo the last action.

event (obsolete)

Superseded by parameters to AuthenticView events.

See also

Declaration: `event` as `DocEditEvent`

Description
The event property holds a `DocEditEvent` object which contains information about the current event.

GetAllowedElements (obsolete)

Superseded by AuthenticRange.CanPerformActionWith

AuthenticRange now supports all functionality of the 'elements' entry helper. Besides querying the elements that can be inserted, appended, etc., you can invoke the action as well. See AuthenticRange.PerformAction for more information.

```
// ----- javascript sample -----
// instead of:
// var 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);
```

See also

Declaration: `GetAllowedElements` (*`nAction`* as `SpyAuthenticElementActions`, *`pStartElement`* as `XMLData`, *`pEndElement`* as `XMLData`, *`pElements`* as Variant)

Description
`GetAllowedElements()` returns the allowed elements for the various actions specified by `nAction`.

JavaScript example:

```
Function GetAllowed()
{
 var objView = Application.ActiveDocument.DocEditView;

 var arrElements = New Array(1);
```


© 2007 Altova GmbH

```
 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
```

© 2007 Altova GmbH

GetNextVisible (obsolete)

Superseded by AuthenticRange.SelectNext

AuthenticRange now supports a wide range of element navigation methods based on document elements like characters, words, tags and many more. Selecting the text passage that represents the content of the next XML element is just one of them.

```
// ----- javascript sample -----
// instead of:
// var 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);
}
```

See also

Declaration: `GetNextVisible` (*pElement* as XMLData) as XMLData

Description
The method gets the next visible XML element in the document.

GetPreviousVisible (obsolete)

Superseded by AuthenticRange.SelectPrevious

AuthenticRange now supports a wide range of element navigation methods based on document elements like characters, words, tags and many more. Selecting the text passage that represents the content of the previous XML element is just one of them.

```
// ----- javascript sample -----
// instead of:
// var objCurrXMLData = ...
// var objXMLData =
Application.ActiveDocument.DocEditView.GetPreviousVisible(objCurrXMLData);
// Application.ActiveDocument.DocEditView.SelectionSet (objXMLData, 0,
objXMLData, -1);
// use now:
var objRange = ...
try
    { objRange.SelectPrevious (spyAuthenticTag).Select(); }
catch (err)
{
    if ((err.number & 0xffff) == 2004)
        MsgBox ("begin of document reached");
    else
        throw (err);
}
```

See also

Declaration: `GetPreviousVisible` (*pElement* as XMLData) as XMLData

Description
The method gets the previous visible XML element in the document.

© 2007 Altova GmbH

IsEditClearEnabled (obsolete)

Superseded by AuthenticRange.IsDeleteEnabled

The IsDeleteEnabled property is now supported for any range of the document, not only the current UI selection.

```
// ----- javascript sample -----
// instead of:
// if (Application.ActiveDocument.DocEditView.IsEditClearEnabled)
//    Application.ActiveDocument.DocEditView.EditClear();
// use now:
var objCurrSelection = Application.ActiveDocument.AuthenticView.Selection;
if (objCurrSelection.IsDeleteEnabled)
    objCurrSelection.Delete();
```

See also

Declaration: `IsEditClearEnabled` as Boolean

Description
True if `EditClear` is possible. See also `Editing operations`.

IsEditCopyEnabled (obsolete)

Superseded by AuthenticRange.IsCopyEnabled

The IsCopyEnabled property is now supported for any range of the document, not only the current UI selection.

```
// ----- javascript sample -----
// instead of:
// if (Application.ActiveDocument.DocEditView.IsEditCopyEnabled)
//    Application.ActiveDocument.DocEditView.EditCopy();
// use now:
var objCurrSelection = Application.ActiveDocument.AuthenticView.Selection;
if (objCurrSelection.IsCopyEnabled)
    objCurrSelection.Copy();
```

See also

Declaration: `IsEditCopyEnabled` as Boolean

Description
True if copy to clipboard is possible. See also `EditCopy` and `Editing operations`.

IsEditCutEnabled (obsolete)

Superseded by AuthenticRange.IsCutEnabled

The IsCutEnabled property is now supported for any range of the document, not only the current UI selection.

```
// ----- javascript sample -----
// instead of:
// if (Application.ActiveDocument.DocEditView.IsEditCutEnabled)
//    Application.ActiveDocument.DocEditView.EditCut();
// use now:
var objCurrSelection = Application.ActiveDocument.AuthenticView.Selection;
if (objCurrSelection.IsCutEnabled)
    objCurrSelection.Cut();
```

See also

Declaration: `IsEditCutEnabled` as Boolean

Description
True if `EditCut` is currently possible. See also `Editing operations`.

IsEditPasteEnabled (obsolete)

Superseded by AuthenticRange.IsPasteEnabled

The IsPasteEnabled property is now supported for any range of the document, not only the current UI selection.

```
// ----- javascript sample -----
// instead of:
// if (Application.ActiveDocument.DocEditView.IsEditPasteEnabled)
//    Application.ActiveDocument.DocEditView.EditPaste();
// use now:
var objCurrSelection = Application.ActiveDocument.AuthenticView.Selection;
if (objCurrSelection.IsPasteEnabled)
    objCurrSelection.Paste();
```

See also

Declaration: `IsEditPasteEnabled` as Boolean

Description
True if `EditPaste` is possible. See also `Editing operations`.

© 2007 Altova GmbH

IsEditRedoEnabled (obsolete)

Superseded by AuthenticView.IsRedoEnabled

```
// ----- javascript sample -----
// instead of:
// if ( Application. ActiveDocument. DocEditView. IsEditRedoEnabled)
//    Application. ActiveDocument. DocEditView. EditRedo( );
// use now:
if ( Application. ActiveDocument. AuthenticView. IsRedoEnabled)
    Application. ActiveDocument. AuthenticView. Redo( );
```

See also

Declaration: `IsEditRedoEnabled` as Boolean

Description
True if `EditRedo` is currently possible. See also `Editing operations`.

IsEditUndoEnabled (obsolete)

Superseded by AuthenticView.IsUndoEnabled

```
// ----- javascript sample -----
// instead of:
// if ( Application. ActiveDocument. DocEditView. IsEditUndoEnabled)
//    Application. ActiveDocument. DocEditView. EditUndo( );
// use now:
if ( Application. ActiveDocument. AuthenticView. IsUndoEnabled)
    Application. ActiveDocument. AuthenticView. Undo( );
```

See also

Declaration: `IsEditUndoEnabled` as Boolean

Description
True if `EditUndo` is possible. See also `Editing operations`.

IsRowAppendEnabled (obsolete)

Superseded by AuthenticRange.IsInDynamicTable

The operations 'insert', 'append', 'delete' and 'duplicate' row are available whenever the selection is inside a dynamic table.

```
// ----- javascript sample -----
// instead of:
// if ( Application.ActiveDocument.DocEditView.IsRowAppendEnabled)
//    Application.ActiveDocument.DocEditView.RowAppend();
// use now:
if (Application.ActiveDocument.AuthenticView.Selection.IsInDynamicTable())
    Application.ActiveDocument.AuthenticView.Selection.AppendRow();
```

See also

Declaration: `IsRowAppendEnabled` as Boolean

Description
True if `RowAppend` is possible. See also `Row operations`.

IsRowDeleteEnabled (obsolete)

Superseded by AuthenticRange.IsInDynamicTable

The operations 'insert', 'append', 'delete' and 'duplicate' row are available whenever the selection is inside a dynamic table.

```
// ----- javascript sample -----
// instead of:
// if ( Application.ActiveDocument.DocEditView.IsRowDeleteEnabled)
//    Application.ActiveDocument.DocEditView.Rowdelete();
// use now:
if (Application.ActiveDocument.AuthenticView.Selection.IsInDynamicTable())
    Application.ActiveDocument.AuthenticView.Selection.DeleteRow();
```

See also

Declaration: `IsRowDeleteEnabled` as Boolean

Description
True if `RowDelete` is possible. See also `Row operations`.

© 2007 Altova GmbH

IsRowDuplicateEnabled (obsolete)

Superseded by AuthenticRange.IsInDynamicTable

The operations 'insert', 'append', 'delete' and 'duplicate' row are available whenever the selection is inside a dynamic table.

```
// ----- javascript sample -----
// instead of:
// if ( Application.ActiveDocument.DocEditView.IsRowDuplicateEnabled)
//    Application.ActiveDocument.DocEditView.RowDuplicate();
// use now:
if ( Application.ActiveDocument.AuthenticView.Selection.IsInDynamicTable())
    Application.ActiveDocument.AuthenticView.Selection.DuplicateRow();
```

See also

Declaration: `IsRowDuplicateEnabled` as Boolean

Description
True if `RowDuplicate` is currently possible. See also `Row operations`.

IsRowInsertEnabled (obsolete)

Superseded by AuthenticRange.IsInDynamicTable

The operations 'insert', 'append', 'delete' and 'duplicate' row are available whenever the selection is inside a dynamic table.

```
// ----- javascript sample -----
// instead of:
// if ( Application.ActiveDocument.DocEditView.IsRowInsertEnabled)
//    Application.ActiveDocument.DocEditView.RowInsert();
// use now:
if ( Application.ActiveDocument.AuthenticView.Selection.IsInDynamicTable())
    Application.ActiveDocument.AuthenticView.Selection.InsertRow();
```

See also

Declaration: `IsRowInsertEnabled` as Boolean

Description
True if `RowInsert` is possible. See also `Row operations`.

IsRowMoveDownEnabled (obsolete)

Superseded by AuthenticRange.IsLastRow

```
// ----- javascript sample -----
// instead of:
// if (Application.ActiveDocument.OldAuthenticView.IsRowMoveDownEnabled)
//    Application.ActiveDocument.DocEditView.RowMoveDown();
// use now:
if (!Application.ActiveDocument.AuthenticView.Selection.IsLastRow)
    Application.ActiveDocument.AuthenticView.Selection.MoveRowDown();
```

See also

Declaration: `IsRowMoveDownEnabled` as Boolean

Description
True if `RowMoveDown` is currently possible. See also `Row operations`.

IsRowMoveUpEnabled (obsolete)

Superseded by AuthenticRange.IsFirstRow

```
// ----- javascript sample -----
// instead of:
// if (Application.ActiveDocument.DocEditView.IsRowMoveUpEnabled)
//    Application.ActiveDocument.DocEditView.RowMoveUp();
// use now:
if (!Application.ActiveDocument.AuthenticView.Selection.IsFirstRow)
    Application.ActiveDocument.AuthenticView.Selection.MoveRowUp();
```

See also

Declaration: `IsRowMoveUpEnabled` as Boolean

Description
True if `RowMoveUp` is possible. See also `Row operations`.

© 2007 Altova GmbH

IsTextStateApplied (obsolete)

Superseded by AuthenticRange.IsTextStateApplied

```
// ----- 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");
```

See also

Declaration: `IsTextStateApplied` (*elementName* as String) as Boolean

Description
Checks to see if the it the text state has already been applied. Common examples for the parameter `elementName` would be strong and italic.

IsTextStateEnabled (obsolete)

Superseded by AuthenticRange.CanPerformAction

Use spyAuthenticApply for the eAction parameter. The CanPerformAction method allows to operate on any range of the document, not only the current UI selection.

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.DocEditView.IsTextStateEnabled ("bold");
// use now:
if (Application.ActiveDocument.AuthenticView.Selection.CanPerformAction
(spyAuthenticApply, "bold"))
    ... // e.g. enable 'bold' button
```

See also

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.

LoadXML (obsolete)

Superseded by AuthenticView.AsXMLString

AuthenticView now supports the property AsXMLString that can be used to directly access and replace the document content as an XMLString.

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.DocEditView.LoadXML (strDocAsXMLString);
// use now:
try
    { Application.ActiveDocument.AuthenticView.AsXMLString =
strDocAsXMLString; }
catch (err)
    { MsgBox ("Error: invalid XML string"); }
```

See also

Declaration: `LoadXML` (*xmlString* as String)

Description
Loads the current XML document with the XML string applied. The new content is displayed immediately.
The *xmlString* parameter must begin with the XML declaration, e.g.,
objPlugIn.LoadXML("<?xml version='1.0' encoding='UTF-8'?><root></root>");

MarkUpView (obsolete)

Superseded by AuthenticView.MarkupVisibility

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.DocEditView.MarkuUpView = 2;
// use now:
Application.ActiveDocument.AuthenticView.MarkupVisibility =
spyAuthenticMarkupLarge;
```

See also

Declaration: `MarkUpView` (*kind* as long)

Description
By default the document displayed is using HTML techniques. But sometimes it is desirable to show the editing tags. Using this method it is possible to display three different types of markup tags:

0	hide the markup tags
2	show the large markup tags
3	show the mixed markup tags.

© 2007 Altova GmbH

RowAppend (obsolete)

Superseded by AuthenticRange.AppendRow

The table operations of AuthenticRange now allow to manipulate any table in the current document independent of the current UI selection.

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.DocEditView.RowAppend();
// use now:
if (! Application.ActiveDocument.AuthenticView.Selection.AppendRow())
    MsgBox ("Error: can't append row");
```

See also

Declaration: `RowAppend`

Description
Appends a row at the current position.

See also Row operations.

RowDelete (obsolete)

Superseded by AuthenticRange.DeleteRow

The table operations of AuthenticRange now allow to manipulate any table in the current document independent of the current UI selection.

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.DocEditView.RowDelete();
// use now:
if (! Application.ActiveDocument.AuthenticView.Selection.DeleteRow())
    MsgBox ("Error: can't delete row");
```

See also

Declaration: `RowDelete`

Description
Deletes the currently selected row(s).

See also Row operations.

© 2007 Altova GmbH

RowDuplicate (obsolete)

Superseded by AuthenticRange.DuplicateRow

The table operations of AuthenticRange now allow to manipulate any table in the current document independent of the current UI selection.

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.DocEditView.RowDuplicate();
// use now:
if (! Application.ActiveDocument.AuthenticView.Selection.DuplicateRow())
    MsgBox ("Error: can't duplicate row");
```

See also

Declaration: `RowDuplicate`

Description
The method duplicates the currently selected rows.

See also Row operations.

RowInsert (obsolete)

Superseded by AuthenticRange.InsertRow

The table operations of AuthenticRange now allow to manipulate any table in the current document independent of the current UI selection.

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.DocEditView.RowInsert();
// use now:
if (! Application.ActiveDocument.AuthenticView.Selection.InsertRow())
    MsgBox ("Error: can't insert row");
```

See also

Declaration: `RowInsert`

Description
Inserts a new row immediately above the current selection.

See also Row operations.

RowMoveDown (obsolete)

Superseded by AuthenticRange.MoveRowDown

The table operations of AuthenticRange now allow to manipulate any table in the current document independent of the current UI selection.

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.DocEditView.RowMoveDown();
// use now:
if (! Application.ActiveDocument.AuthenticView.Selection.MoveRowDown())
    MsgBox ("Error: can't move row down");
```

See also

Declaration: `RowMoveDown`

Description
Moves the current row one position down.

See also Row operations.

RowMoveUp (obsolete)

Superseded by AuthenticRange.MoveRowUp

The table operations of AuthenticRange now allow to manipulate any table in the current document independent of the current UI selection.

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.DocEditView.RowAppend();
// use now:
if (! Application.ActiveDocument.AuthenticView.Selection.MoveRowUp())
    MsgBox ("Error: can't move row up");
```

See also

Declaration: `RowMoveUp`

Description
Moves the current row one position up.

See also Row operations.

© 2007 Altova GmbH

SaveXML (obsolete)

Superseded by AuthenticView.AsXMLString

AuthenticView now supports the property XMLString that can be used to directly access and replace the document content as an XMLString.

```
// ----- javascript sample -----
// instead of:
// var strDocAsXMLString = Application.ActiveDocument.DocEditView.SaveXML();
// use now:
try
{
    var strDocAsXMLString =
Application.ActiveDocument.AuthenticView.AsXMLString;
    ... // do something here
}
catch (err)
    { MsgBox ("Error: invalid XML string"); }
```

See also

Declaration: `SaveXML` as String

Return Value
XML structure as string

Description
Saves the current XML data to a string that is returned to the caller.

SelectionMoveTabOrder (obsolete)

Superseded by AuthenticRange.SelectNext

AuthenticRange now supports a wide range of element navigation methods based on document elements like characters, words, tags and many more. Selecting the next paragraph is just one of them, and navigation is not necessarily bound to the current UI selection.

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.DocEditView.SelectionMoveTabOrder(true, true);
// use now:
Application.ActiveDocument.AuthenticView.Selection.SelectNext
(spyAuthenticParagraph).Select();
// to append a row to a table use AuthenticRange.AppendRow
```

See also

Declaration: `SelectionMoveTabOrder` (*bForward* as Boolean, *bTag* as Boolean)

Description

`SelectionMoveTabOrder()` moves the current selection forwards or backwards.

If `bTag` is false and the current selection is at the last cell of a table a new line will be added.

© 2007 Altova GmbH

SelectionSet (obsolete)

Superseded by AuthenticRange.FirstXMLData and related properties

AuthenticRange supports navigation via XMLData elements as well as navigation by document elements (e.g. characters, words, tags) or text cursor positions.

```
// ----- javascript sample -----
// instead of:
// if (! Application.ActiveDocument.DocEditView.SelectionSet(varXMLData1, 0,
varXMLData2, -1))
//    MsgBox ("Error: invalid data position");
// use now:
try
{
    var objSelection = Application.ActiveDocument.AuthenticView.Selection;
    objSelection.FirstXMLData = varXMLData1;
    objSelection.FirstXMLdataOffset = 0;
    objSelection.LastXMLData = varXMLData2;
    objSelection.LastXMLDataOffset = -1;
    objSelection.Select();
}
catch (err)
    { MsgBox ("Error: invalid data position"); }
// to select all text between varXMLData1 and varXMLdata2, inclusive
```

See also

Declaration: `SelectionSet` (*`pStartElement`* as `XMLData`, *`nStartPos`* as long, *`pEndElement`* as `XMLData`, *`nEndPos`* as long) as Boolean

Description

Use `SelectionSet()` to set a new selection in the Authentic. Its possible to set `pEndElement` to null (nothing) if the selection should be just over one (`pStartElement`) XML element.

XMLRoot (obsolete)

Superseded by AuthenticView.XMLDataRoot

```
// ----- javascript sample -----
// instead of:
// var objXMLData = Application.ActiveDocument.DocEditView.XMLRoot;
// use now:
var objXMLData = Application.ActiveDocument.AuthenticView.XMLDataRoot;
```

See also

Declaration: `XMLRoot` as `XMLData`

Description

`XMLRoot` is the parent element of the currently displayed XML structure. Using the `XMLData` interface you have full access to the complete content of the file.

See also Using XMLData for more information.

© 2007 Altova GmbH

4.5 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.

4.5.1 SPYAttributeTypeDefinition

Description
Attribute type definition that can be selected for generation of Sample XML.
This type is used with the method GenerateDTDOrSchema and GenerateDTDOrSchemaEx.

Possible values:

spyMergedGlobal	= 0
spyDistinctGlobal	= 1
spyLocal	= 2

4.5.2 SPYAuthenticActions

Description
Actions that can be performed on AuthenticRange objects.

Possible values:

spyAuthenticInsertAt	= 0
spyAuthenticApply	= 1
spyAuthenticClearSurr	= 2
spyAuthenticAppend	= 3
spyAuthenticInsertBefore	= 4
spyAuthenticRemove	= 5

4.5.3 SPYAuthenticDocumentPosition

Description
Relative and absolute positions used for navigating with AuthenticRange objects.

Possible values:

spyAuthenticDocumentBegin	= 0
spyAuthenticDocumentEnd	= 1
spyAuthenticRangeBegin	= 2
spyAuthenticRangeEnd	= 3

4.5.4 SPYAuthenticElementActions

Description
Actions that can be used with GetAllowedElements (superseded by `AuthenticRange.CanPerformActionWith`).

Possible values:

k_ActionInsertAt	= 0
k_ActionApply	= 1
k_ActionClearSurr	= 2
k_ActionAppend	= 3
k_ActionInsertBefore	= 4
k_ActionRemove	= 5

© 2007 Altova GmbH

4.5.5 SPYAuthenticElementKind

Description
Enumeration of the different kinds of elements used for navigation and selection within the AuthenticRange and AuthenticView objects.

Possible values:

spyAuthenticChar	= 0
spyAuthenticWord	= 1
spyAuthenticLine	= 3
spyAuthenticParagraph	= 4
spyAuthenticTag	= 6
spyAuthenticDocument	= 8
spyAuthenticTable	= 9
spyAuthenticTableRow	= 10
spyAuthenticTableColumn	= 11

4.5.6 SPYAuthenticMarkupVisibility

Description
Enumeration values to customize the visibility of markup with MarkupVisibility.

Possible values:

spyAuthenticMarkupHidden	= 0
spyAuthenticMarkupSmall	= 1
spyAuthenticMarkupLarge	= 2
spyAuthenticMarkupMixed	= 3

4.5.7 SPYDatabaseKind

Description
Values to select different kinds of databases for import. See DatabaseConnection.DatabaseKind for its use.

Possible values:

spyDB_Access	= 0
spyDB_SQLServer	= 1
spyDB_Oracle	= 2
spyDB_Sybase	= 3
spyDB_MySQL	= 4
spyDB_DB2	= 5
spyDB_Other	= 6
spyDB_Unspecified	= 7

4.5.8 SPYDialogAction

Description
Values to simulate different interactions on dialogs. See Dialogs for all dialogs available.

Possible values:

spyDialogOK	= 0	//simulate click on OK button
spyDialogCancel	= 1	//simulate click on Cancel button
spyDialogUserInput	= 2	// show dialog and allow user interaction

4.5.9 SPYDOMType

Description
Enumeration values to parameterize generation of C++ code from schema definitions.

Possible values:

spyDOMType_msxml4	= 0
spyDOMType_xerces	= 1

4.5.10 SPYDTDSchemaFormat

Description
Enumeration to identify the different schema formats.

Possible values:

spyDTD	= 0
spyW3C	= 1

4.5.11 SPYEncodingByteOrder

Description
Enumeration values to specify encoding byte ordering for text import and export.

Possible values:

spyNONE	= 0
spyLITTLE_ENDIAN	= 1
spyBIG_ENDIAN	= 2

4.5.12 SPYExportNamespace

Description
Enumeration type to configure handling of namespace identifiers during export.

Possible values:

spyNoNamespace	= 0
spyReplaceColonWithUnderscore	= 1

4.5.13 SPYFindInFilesSearchLocation

Description
The different locations where a search can be performed. This type is used with the FindInFilesDlg dialog.

Possible values:

spyFindInFiles_Documents	= 0
spyFindInFiles_Project	= 1
spyFindInFiles_Folder	= 2

4.5.14 SPYFrequentElements

Description
Enumeration value to parameterize schema generation.

Possible values:

© 2007 Altova GmbH

spyGlobalElements = 0
spyGlobalComplexType = 1

4.5.15 SPYImageKind

Description
Enumeration values to parameterize image type of the generated documentation. These values are used in SchemaDocumentationDialog.DiagramFormat and WSDLDocumentationDlg.DiagramFormat.

Possible values:
spyImageType_PNG = 0
spyImageType_EMF = 1

4.5.16 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

4.5.17 SPYLibType

Description
Enumeration values to parameterize generation of C++ code from schema definitions.

Possible values:
spyLibType_static = 0
spyLibType_dll = 1

4.5.18 SPYLoading

Description
Enumeration values to define loading behaviour of URL files.

Possible values:
spyUseCacheProxy = 0
spyReload = 1

4.5.19 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	
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

4.5.20 SPYNumberDateTimeFormat

Description
Enumeration value to configure database connections.

Possible values:

spySystemLocale	= 0
spySchemaCompatible	= 1

4.5.21 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

4.5.22 SPYProjectItemTypes

Description
Enumeration values to identify the different elements in project item lists. See SpyProjectItem.ItemType.

© 2007 Altova GmbH

Possible values:

spyUnknownItem	= 0
spyFileItem	= 1
spyFolderItem	= 2
spyURLItem	= 3

4.5.23 SPYProjectType

Description
Enumeration values to parameterize generation of C# code from schema definitions.

Possible values:

spyVisualStudioProject	= 0
spyVisualStudio2003Project	= 1
spyBorlandProject	= 2
spyMonoMakefile	= 3
spyVisualStudio2005Project	= 4

4.5.24 SPYSampleXMLGenerationOptimization

Description
Specify the elements that will be generated in the Sample XML.
This enumeration is used in GenerateSampleXMLDlg.

Possible values:

spySampleXMLGen_Optimized	= 0
spySampleXMLGen_NonMandatoryElements	= 1
spySampleXMLGen_Everything	= 2

4.5.25 SPYSampleXMLGenerationSchemaOrDTDAssignment

Description
Specifies what kind of reference to the schema/DTD should be added to the generated Sample XML.
This enumeration is used in GenerateSampleXMLDlg.

Possible values:

spySampleXMLGen_AssignRelatively	= 0
spySampleXMLGen_AssignAbsolutely	= 1
spySampleXMLGen_DoNotAssign	= 2

4.5.26 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

4.5.27 SPYSchemaDocumentationFormat

Description
Enumeration values to parameterize generation of schema documentation. These values are used in SchemaDocumentationDialog.OutputFormat and WSDLDocumentationDlg.OutputFormat.

Possible values:

spySchemaDoc_HTML	= 0
spySchemaDoc_MSWord	= 1
spySchemaDoc_RTF	= 2

4.5.28 SPYTextDelimiters

Description
Enumeration values to specify text delimiters for text export.

Possible values:

spyTabulator	= 0
spySemicolon	= 1
spyComma	= 2
spySpace	= 3

4.5.29 SPYTextEnclosing

Description
Enumeration value to specify text enclosing characters for text import and export.

Possible values:

spyNoEnclosing	= 0
spySingleQuote	= 1
spyDoubleQuote	= 2

4.5.30 SPYTypeDetection

Description
Enumeration to select how type detection works during GenerateDTDOrSchema and GenerateDTDOrSchemaEx.

Possible values:

spyBestPossible	= 0

© 2007 Altova GmbH

spyNumbersOnly = 1
spyNoDetection = 2

4.5.31 SPYURLTypes

Description
Enumeration to specify different URL types.

Possible values:
spyURLTypeAuto = -1
spyURLTypeXML = 0
spyURLTypeDTD = 1

4.5.32 SPYViewModes

Description
Enumeration values that define the different view modes for XML documents. The mode *spyViewContent(4)* identifies the mode that was intermediately called DocEdit mode and is now called Authentic mode. The mode *spyViewWSDL* identifies a mode which is mapped to the schema view on the GUI but distinguished internally.

Possible values:
spyViewGrid = 0
spyViewText = 1
spyViewBrowser = 2
spyViewSchema = 3
spyViewContent = 4 // obsolete
spyViewAuthentic = 4
spyViewWSDL = 5

4.5.33 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

4.5.34 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

© 2007 Altova GmbH

5 Using the XMLSpy API with Java

The XMLSpy API has an interface built up of Java classes, each of which corresponds to an object in the XMLSpy API. Developers can use these Java classes to interact with the COM API. These classes are listed below and described in subsequent sections. For a description of the XMLSpy API objects themselves, see the XMLSpy API documentation. Bear in mind that some API features are only available in scripting environments; these have therefore not been ported to Java.

Java classes

SpyApplication
  SpyProject
    SpyProjectItems
      SpyProjectItem
  SpyDocuments
    SpyDoc
      SpyAuthenticView
        SpyAuthenticRange
      SpyDocEditView
        SpyDocEditSelection
      SpyGridView
      SpyXMLData
  SpyDialogs
    SpyCodeGeneratorDlg
    SpyFileSelectionDlg
    SpySchemaDocumentationDlg
  SpyDatabaseConnection
  SpyElementList
  SpyElementListItem
  SpyExportSettings
  SpyTextImportExportSettings

Implementation of COM properties in Java

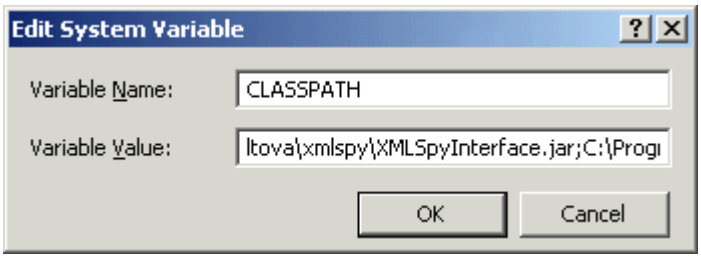
Properties in Java have been defined to include both a **`set`** and **`get`** method (`set` if it is allowed by the COM implementation). For example, the COM class `Document` contains the **`GridView`** property. In Java the method is called `SpyDoc` and the property is defined as a **`GetGridView`** method.

If you encounter compiling problems, please check the following points:

- The `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 2000 and 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).

© 2007 Altova GmbH

5.1 Sample source code

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 = new SpyApplication();
  app.ShowApplication( true );

  SpyDoc doc = app.GetDocuments().OpenFile("C:\\Program Files\\Altova\\xmlspy
2007\\Examples\\OrgChart.xml", true );

  doc.SwitchViewMode(SPYViewModes.spyViewGrid);
  SpyXMLData oData = doc.GetRootElement();
  SpyXMLData oNewChild = doc.CreateChild(SPYXMLDataKind.spyXMLDataElement);

  oNewChild.SetName( "NewChild" );
  oNewChild.SetTextValue("newVaLuE");
  oData.AppendChild(oNewChild);
  oNewChild.ReleaseInstance();
  oData.ReleaseInstance();


  if ( doc.IsValid( ) == false )
  {
   // is to be expected after above insertion
   System.out.println( "!!!!!! validation error: "+doc.GetErrorString() );
   System.out.println( "!!!!!! validation error: "+doc.GetErrorPos() );
   System.out.println( "!!!!!! validation error: "+doc.GetBadData() );

  }

  doc.ReleaseInstance();
  app.Quit();
  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*

© 2007 Altova GmbH

5.2 SpyApplication

```
public class SpyApplication
{
  public void ReleaseInstance();
  public void ShowApplication( boolean bShow );
  public void Quit ();
  public void AddMacroMenuItem( String sMacro, String sDisplayText );
  public void ClearMacroMenu( );
  public SpyDoc GetActiveDocument( );
  public SpyProject GetCurrentProject( );
  public SpyDocuments GetDocuments( );
  public SpyElementList GetDatabaseImportElementList( SpyDatabaseConnection
  oImportSettings );
  public SpyDatabaseConnection GetDatabaseSettings();
  public SpyElementList GetDatabaseTables( SpyDatabaseConnection
  oImportSettings );
  public SpyExportSettings GetExportSettings();
  public SpyElementList GetTextImportElementList( SpyTextImportExportSettings
  oImportSettings );
  public SpyTextImportExportSettings GetTextImportExportSettings();
  public SpyDoc ImportFromDatabase( SpyDatabaseConnection oImportSettings,
  SpyElementList oElementList );
  public SpyDoc ImportFromSchema( SpyDatabaseConnection oImportSettings,
  String strTable, SpyDoc oSchemaDoc );
  public SpyDoc ImportFromText( SpyTextImportExportSettings oImportSettings,
  SpyElementList oElementList );
  public SpyDoc ImportFromWord( String sFile );
  public void NewProject( String sPath, boolean bDiscardCurrent );
  public void OpenProject( String sPath , boolean bDiscardCurrent, boolean
  bDialog );
  public long ShowForm( String sName );
  public void URLDelete( String sURL, String sUser, String sPassword );
  public void URLMakeDirectory( String sURL, String sUser, String sPassword );

  public int GetWarningNumber();
  public String GetWarningText();

  // since Version 2004R4
  public SpyApplication GetApplication()
  public SpyApplication GetParent()
  public SpyDialogs GetDialogs()
  public boolean GetVisible()
  public void SetVisible( boolean i_bVisibility)
  public long GetWindowHandle()
}
```

5.3 SpyCodeGeneratorDlg

Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

```
// since version 2004R4
public class SpyCodeGeneratorDlg
{
   public SpyApplication GetApplication();
   public SpyDialogs GetParent();
   public long GetProgrammingLanguage();
   public void SetProgrammingLanguage( long i_eVal );
   public String GetTemplateFileName();
   public void SetTemplateFileName( String i_strVal );
   public String GetOutputPath();
   public void SetOutputPath( String i_strVal );
   public long GetOutputPathDialogAction();
   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 );
}
```


© 2007 Altova GmbH

5.4 SpyDatabaseConnection

```
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 )
}
```

5.5 SpyDialogs

```
// Since version 2004R4
public class SpyDialogs
{
  public SpyApplication GetApplication();
  public SpyApplication GetParent();
  public SpyCodeGeneratorDlg GetCodeGeneratorDlg ();
  public SpyFileSelectionDlg GetFileSelectionDlg ();
  public SpySchemaDocumentationDlg GetSchemaDocumentationDlg ();
}
```


© 2007 Altova GmbH

5.6 SpyDoc

```
public class SpyDoc
{
  public void ReleaseInstance();
  public void SetEncoding( String strEncoding );
  public void SetPathName( String strPath );
  public String GetPathName( );
  public String GetTitle( );
  public boolean IsModified( );
  public void Save( );
  public void Close( boolean bDiscardChanges );
  public void UpdateViews( );
  public long GetCurrentViewMode( );
  public boolean SwitchViewMode( long nMode );
  public SpyGridView GetGridView( );
  public void SetActiveDocument( );
  public void StartChanges( );
  public void EndChanges( );
  public void TransformXSL( );
  public void AssignDTD( String sDTDFile, boolean bDialog );
  public void AssignSchema( String sSchemaFile, boolean bDialog );
  public void AssignXSL( String sXSLFile, boolean bDialog );
  public void ConvertDTDOrSchema( long nFormat, long nFrequentElements );
  public SpyXMLData CreateChild( long nKind );
  public void CreateSchemaDiagram( long nKind, String sName, String sFile );
  public SpyDocEditView GetDocEditView();
  public void ExportToDatabase( SpyXMLData oFromChild, SpyExportSettings
  oExportSettings, SpyDatabaseConnection oDatabaseConnection );
  public void ExportToText( SpyXMLData oFromChild, SpyExportSettings
  oExportSettings, SpyTextImportExportSettings oTextSettings );
  public void GenerateDTDOrSchema( long nFormat, int nValuesList, long
  nDetection, long nFrequentElements );
  public SpyElementList GetExportElementList( SpyXMLData oFromChild,
  SpyExportSettings oExportSettings );
  public SpyXMLData GetRootElement();
  public String SaveInString( SpyXMLData oData, boolean bMarked );
  public void SaveToURL( String sUrl, String sUser, String sPassword );
  public String GetErrorString();
  public int GetErrorPos();
  public SpyXMLData GetBadData();
  public boolean IsValid();
  public boolean IsWellFormed( SpyXMLData oData, boolean bWithChildren );

  // Since version 2004R3
  public SpyAuthenticView GetAuthenticView()

  // Since version 2004R4
  public SpyApplication GetApplication();
  public SpyDocuments GetParent();
  public String GetFullName();
  public void SetFullName( String i_strName );
  public String GetName();
  public String GetPath();
  public boolean GetSaved();
  public void SaveAs( String i_strFileNameOrPath );
  public String GetEncoding();
  public SpyXMLData GetDataRoot();
  public void GenerateProgramCode( SpyCodeGeneratorDlg i_dlg );
  public void AssignXSLFO( String i_strFile, boolean i_bUseDialog );
  public void TransformXSLFO ();
```

```
    public void GenerateSchemaDocumentation (  SpySchemaDocumentationDlg i_dlg );
}
```


© 2007 Altova GmbH

5.7 SpyDocuments

```
public class SpyDocuments
{
  public void ReleaseInstance();
  public long Count();
  public SpyDoc GetItem( long nNo );
  public SpyDoc NewFile( String strFile, String strType );
  public SpyDoc NewFileFromText( String nSource, String strType );
  public SpyDoc OpenFile( String sPath, boolean bDialog );
  public SpyDoc OpenURL( String sUrl, long nURLType, long nLoading, String
  sUser, String sPassword );
  public SpyDoc OpenURLDialog( String sURL, long nURLType, long nLoading,
  String sUser, String sPassword )
}
```

5.8 SpyElementList

```
public class SpyElementList
{
  public void ReleaseInstance();
  public long GetCount();
  public SpyElementListItem  GetItem( long nIndex );
  public void RemoveElement( long nIndex );
}
```


© 2007 Altova GmbH

5.9 SpyElementListItem

```
public class SpyElementListItem
{
  public void ReleaseInstance();
  public long GetElementKind();
  public void SetElementKind( long nKind );
  public long GetFieldCount ();
  public String GetName();
  public long GetRecordCount();
}
```

5.10 SpyExportSettings

```
public class SpyExportSettings
{
  public void ReleaseInstance();
  public boolean GetCreateKeys();
  public void SetCreateKeys( boolean bValue );
  public SpyElementList GetElementList();
  public void SetElementList( SpyElementList obj );
  public boolean GetEntitiesToText ();
  public void SetEntitiesToText ( boolean bValue );
  public boolean GetExportAllElements ();
  public void SetExportAllElements ( boolean bValue );
  public boolean GetFromAttributes ();
  public void SetFromAttributes ( boolean bValue );
  public boolean GetFromSingleSubElements ();
  public void SetFromSingleSubElements ( boolean bValue );
  public boolean GetFromTextValues ();
  public void SetFromTextValues ( boolean bValue );
  public boolean GetIndependentPrimaryKey ();
  public void SetIndependentPrimaryKey ( boolean bValue );
  public long GetNamespace ();
  public void SetNamespace ( long nValue );
  public int GetSubLevelLimit ();
  public void SetSubLevelLimit ( int nValue );
}
```


© 2007 Altova GmbH

5.11 SpyFileSelectionDlg

```
// Since version 2004R4
public class SpyFileSelectionDlg
{
  public SpyApplication GetApplication();
  public SpyDialogs GetParent();
  public String GetFullName();
  public void SetFullName( String i_strName );
  public long GetDialogAction();
  public void SetDialogAction( long i_eAction );
}
```

5.12 SpyGridView

```
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 );
}
```


© 2007 Altova GmbH

5.13 SpyProject

```
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 );
}
```

5.14 SpyProjectItem

```
public class SpyProjectItem
{
  public void ReleaseInstance();
  public SpyProjectItems GetChildItems ();
  public String GetFileExtensions();
  public void SetFileExtensions( String sExtensions );
  public long GetItemType();
  public String GetName();
  public SpyDoc Open();
  public SpyProjectItem GetParentItem();
  public String GetPath();
  public String GetValidateWith();
  public void SetValidateWith( String sVal );
  public String GetXMLForXSLTransformation();
  public void SetXMLForXSLTransformation( String sVal );
  public String GetXSLForXMLTransformation();
  public void SetXSLForXMLTransformation( String sVal );
  public String GetXSLTransformationFileExtension();
  public void SetXSLTransformationFileExtension( String sVal );
  public String GetXSLTransformationFolder();
  public void SetXSLTransformationFolder( String sVal );
}
```


© 2007 Altova GmbH

5.15 SpyProjectItems

```
public class SpyProjectItems
{
  public void ReleaseInstance();
  public void AddFile( String sPath );
  public void AddFolder( String sName );
  public void AddURL( String sURL, long nURLType, String sUser, String
  sPassword, boolean bSave );
  public long Count();
  public SpyProjectItem GetItem( long nNumber );
  public void RemoveItem( SpyProjectItem oItemToRemove );
}
```

5.16 SpySchemaDocumentationDlg

```
// Since version 2004R4
public class SpySchemaDocumenttaionDlg
{
   public SpyApplication GetApplication();
   public SpyDialogs GetParent();

   public String GetOutputFile();
   public void SetOutputFile( String i_strVal );
   public long GetOutputFormat();
   public void SetOutputFormat( long i_eVal );

   public boolean GetShowResult();
   public void SetShowResult( boolean i_bVal );
   public long GetOptionsDialogAction();
   public void SetOptionsDialogAction( long i_eVal );
   public long GetOutputFileDialogAction();
   public void SetOutputFileDialogAction( long i_eVal );
   public boolean GetShowProgressBar();
   public void SetShowProgressBar( boolean i_bVal );

   public void IncludeAll( boolean i_bInclude );
   public boolean GetIncludeIndex();
   public void SetIncludeIndex( boolean i_bVal );
   public boolean GetIncludeGlobalElements();
   public void SetIncludeGlobalElements( boolean i_bVal );
   public boolean GetIncludeLocalElements();
   public void SetIncludeLocalElements( boolean i_bVal );
   public boolean GetIncludeGroups();
   public void SetIncludeGroups( boolean i_bVal );
   public boolean GetIncludeComplexTypes();
   public void SetIncludeComplexTypes( boolean i_bVal );
   public boolean GetIncludeSimpleTypes();
   public void SetIncludeSimpleTypes( boolean i_bVal );
   public boolean GetIncludeAttributeGroups();
   public void SetIncludeAttributeGroups( boolean i_bVal );
   public boolean GetIncludeRedefines();
   public void SetIncludeRedefines( boolean i_bVal );

   public void AllDetails( boolean i_bDetailsOn );
   public boolean GetShowDiagram();
   public void SetShowDiagram( boolean i_bVal );
   public boolean GetShowNamespace();
   public void SetShowNamespace( boolean i_bVal );
   public boolean GetShowType();
   public void SetShowType( boolean i_bVal );
   public boolean GetShowChildren();
   public void SetShowChildren( boolean i_bVal );
   public boolean GetShowUsedBy();
   public void SetShowUsedBy( boolean i_bVal );
   public boolean GetShowProperties();
   public void SetShowProperties( boolean i_bVal );
   public boolean GetShowSingleFacets();
   public void SetShowSingleFacets( boolean i_bVal );
   public boolean GetShowPatterns();
   public void SetShowPatterns( boolean i_bVal );
   public boolean GetShowEnumerations();
   public void SetShowEnumerations( boolean i_bVal );
   public boolean GetShowAttributes();
   public void SetShowAttributes( boolean i_bVal );
```

 © 2007 Altova GmbH

```
   public boolean GetShowIdentityConstraints();
   public void SetShowIdentityConstraints( boolean i_bVal );
   public boolean GetShowAnnotations();
   public void SetShowAnnotations( boolean i_bVal );
   public boolean GetShowSourceCode();
   public void SetShowSourceCode( boolean i_bVal );
}
```

5.17 SpyTextImportExportSettings

```
public class SpyTextImportExportSettings
{
  public void ReleaseInstance();
  public String GetDestinationFolder ();
  public void SetDestinationFolder ( String sVal );
  public long GetEnclosingCharacter ();
  public void SetEnclosingCharacter ( long nEnclosing );
  public String GetEncoding ();
  public void SetEncoding ( String sVal );
  public long GetEncodingByteOrder();
  public void SetEncodingByteOrder( long nByteOrder );
  public long GetFieldDelimiter();
  public void SetFieldDelimiter( long nDelimiter );
  public String GetFileExtension ();
  public void SetFileExtension ( String sVal );
  public boolean GetHeaderRow();
  public void SetHeaderRow( boolean bVal );
  public String GetImportFile();
  public void SetImportFile( String sVal );
}
```


© 2007 Altova GmbH

5.18 SpyXMLData

```
public class SpyXMLData
{
  public void ReleaseInstance();
  public void AppendChild( SpyXMLData oNewData );
  public void EraseAllChildren();
  public void EraseCurrentChild();
  public SpyXMLData GetCurrentChild();
  public SpyXMLData GetFirstChild( long nKind );
  public SpyXMLData GetNextChild();
  public boolean GetHasChildren();
  public void InsertChild( SpyXMLData oNewData );
  public boolean IsSameNode( SpyXMLData oToComp);
  public long GetKind();
  public boolean GetMayHaveChildren ();
  public String GetName();
  public void SetName( String sValue );
  public SpyXMLData GetParent();
  public String GetTextValue();
  public void SetTextValue ( String sValue );
}
```

5.19 Authentic

5.19.1 SpyAuthenticRange

```
// Since version 2004R3
public class SpyAuthenticRange
{
    public void ReleaseInstance();
    public SpyApplication GetApplication();
    public SpyAuthenticView GetParent();
    public SpyAuthenticRange GotoNext(long eKind);
    public SpyAuthenticRange GotoPrevious(long eKind);
    public void Select();
    public long GetFirstTextPosition();
    public void SetFirstTextPosition (long nTextPosition);
    public long GetLastTextPosition();
    public void SetLastTextPosition (long nTextPosition);
    public String GetText();
    public void SetText (String strText);
    public boolean PerformAction(long eAction, String strElementName);
    public boolean CanPerformAction(long eAction, String strElementName);
    public String[] CanPerformActionWith (long eAction);
    public SpyAuthenticRange GoTo(long eKind, long nCount, long nFrom);
    public SpyAuthenticRange SelectNext (long eKind);
    public SpyAuthenticRange SelectPrevious (long eKind);
    public SpyAuthenticRange MoveBegin (long eKind, long nCount);
    public SpyAuthenticRange MoveEnd (long eKind, long nCount);
    public SpyAuthenticRange ExpandTo (long eKind);
    public SpyAuthenticRange CollapsToBegin ();
    public SpyAuthenticRange CollapsToEnd ();
    public SpyAuthenticRange GotoNextCursorPosition ();
    public SpyAuthenticRange GotoPreviousCursorPosition ();
    public boolean IsEmpty ();
    public boolean IsEqual (SpyAuthenticRange ipCmp);
    public SpyAuthenticRange Clone ();
    public SpyAuthenticRange SetFromRange (SpyAuthenticRange ipSrc);
    public boolean Delete ();
    public boolean Cut ();
    public boolean Copy ();
    public boolean Paste ();
    public SpyXMLData GetFirstXMLData ();
    public void SetFirstXMLData (SpyXMLData objXMLDataPtr);
    public long GetFirstXMLDataOffset ();
    public void SetFirstXMLDataOffset (long nOffset);
    public SpyXMLData GetLastXMLData ();
    public void SetLastXMLData (SpyXMLData objXMLDataPtr);
    public long GetLastXMLDataOffset ();
    public void SetLastXMLDataOffset (long nOffset);
    public String[] GetElementHierarchy ();
    public String[] GetElementAttributeNames (String strElementName);
    public boolean HasElementAttribute (String strElementName, String
strAttributeName);
    public String GetElementAttributeValue (String strElementName, String
strAttributeName);
    public void SetElementAttributeValue (String strElementName, String
strAttributeName, String strNewValue);
    public String[] GetEntityNames ();
    public void InsertEntity (String strEntityName);
    public boolean IsInDynamicTable ();
    public boolean AppendRow ();
    public boolean InsertRow ();
```

 © 2007 Altova GmbH

```
    public boolean DuplicateRow ();
    public boolean DeleteRow ();
    public boolean MoveRowUp ();
    public boolean MoveRowDown ();

    // Since version 2004R4
    public boolean IsCopyEnabled();
    public boolean IsCutEnabled();
    public boolean IsPasteEnabled();
    public boolean IsDeleteEnabled();
    public boolean IsTextStateApplied(String i_strElementName);
    public boolean IsFirstRow();
    public boolean IsLastRow();
}
```

5.19.2 SpyAuthenticView

```
// Since version 2004R3
public class SpyAuthenticView
{
    public void ReleaseInstance();
    public SpyApplication GetApplication();
    public SpyDoc GetParent();
    public SpyAuthenticRange GetSelection();
    public void SetSelection (SpyAuthenticRange obj);
    public SpyAuthenticRange GetDocumentBegin();
    public SpyAuthenticRange GetDocumentEnd();
    public SpyAuthenticRange GetWholeDocument();
    public long GetMarkupVisibility();
    public void SetMarkupVisibility (long eSpyAuthenticMarkupVisibility);
    public SpyAuthenticRange GoTo(long eKind, long nCount, long nFrom);
    public void Print(boolean bWithPreview, boolean bPromptUser);
    public boolean Undo();
    public boolean Redo();
    public void UpdateXMLInstanceEntities();

    // Since version 2004R4
    public String GetAsXMLString();
    public void SetAsXMLString(String i_strXML);
    public SpyXMLData GetXMLDataRoot();
    public boolean IsUndoEnabled ();
    public boolean IsRedoEnabled ();
}
```

5.19.3 SpyDocEditSelection

```
public class SpyDocEditSelection
{
  public void ReleaseInstance();
  public SpyXMLData GetEnd();
  public long GetEndTextPosition();
  public SpyXMLData GetStart();
  public long GetStartTextPosition();
}
```

5.19.4 SpyDocEditView

```
public class SpyDocEditView
{
```

```
    public void ReleaseInstance();
    public void ApplyTextState( String sElementName );
    public SpyDocEditSelection GetCurrentSelection();
    public void EditClear();
    public void EditCopy();
    public void EditCut();
    public void EditPaste();
    public void EditRedo();
    public void EditSelectAll();
    public void EditUndo();
    public SpyXMLData GetNextVisible( SpyXMLData oElement );
    public SpyXMLData GetPreviousVisible( SpyXMLData oElement );
    public boolean GetIsEditClearEnabled ();
    public boolean GetIsEditCopyEnabled ();
    public boolean GetIsEditCutEnabled ();
    public boolean GetIsEditPasteEnabled ();
    public boolean GetIsEditRedoEnabled ();
    public boolean GetIsEditUndoEnabled ();
    public boolean GetIsRowAppendEnabled ();
    public boolean GetIsRowDeleteEnabled ();
    public boolean GetIsRowDuplicateEnabled();
    public boolean GetIsRowInsertEnabled();
    public boolean GetIsRowMoveDownEnabled();
    public boolean GetIsRowMoveUpEnabled();
    public boolean IsTextStateApplied( String sElementName );
    public boolean IsTextStateEnabled( String sElementName );
    public void LoadXML( String sXML );
    public void MarkUpView( long nKind );
    public void RowAppend();
    public void RowDelete();
    public void RowDuplicate();
    public void RowInsert();
    public void RowMoveDown();
    public void RowMoveUp();
    public String SaveXML();
    public void SelectionMoveTabOrder( boolean bForward, boolean bTag );
    public boolean SelectionSet( SpyXMLData oStart, long nStartPos,SpyXMLData
    oEndElement, long nEndPos );
    public SpyXMLData GetXMLRoot();
    public String[] GetAllowedElements( long nAction, SpyXMLData oStartPtr,
    SpyXMLData oEndPtr );
}
```

 © 2007 Altova GmbH

5.20 Predefined constants

This section lists all classes that define the predefined constants used by the Java interface.

5.20.1 SPYAuthenticActions

```
public class SPYAuthenticActions
{
   public final static long                    = 0;
   spyAuthenticInsertAt
   public final static long spyAuthenticApply= 1;
   public final static long                    = 2;
   spyAuthenticClearSurr
   public final static long                    = 3;
   spyAuthenticAppend
   public final static long                    = 4;
   spyAuthenticInsertBefore
   public final static long                    = 5;
   spyAuthenticRemove
}
```

5.20.2 SPYAuthenticDocumentPosition

```
public class SPYAuthenticDocumentPosition
{
   public final static long                    = 0;
   spyAuthenticDocumentBegin
   public final static long                    = 1;
   spyAuthenticDocumentEnd
   public final static long                    = 2;
   spyAuthenticRangeBegin
   public final static long                    = 3;
   spyAuthenticRangeEnd
}
```

5.20.3 SPYAuthenticElementKind

```
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
```

```
    public final static long          = 10;
    spyAuthenticTableRow
    public final static long          = 11;
    spyAuthenticTableColumn
}
```

5.20.4 SPYAuthenticMarkupVisibility

```
public class SPYAuthenticMarkupVisibility
{
    public final static long                  = 0;
    spyAuthenticMarkupHidden
    public final static long                  = 1;
    spyAuthenticMarkupSmall
    public final static long                  = 2;
    spyAuthenticMarkupLarge
    public final static long                  = 3;
    spyAuthenticMarkupMixed
}
```

5.20.5 SPYDatabaseKind

```
public class SPYLoading
{
    public final static long           = 0;
    spyDB_Access
    public final static long           = 1;
    spyDB SQLServer
    public final static long           = 2;
    spyDB_Oracle
    public final static long           = 3;
    spyDB_Sybase
    public final static long           = 4;
    spyDB MySQL
    public final static long spyDB_DB2 = 5;
    public final static long           = 6;
    spyDB Other
    public final static long           = 7;
    spyDB_Unspecified
}
```

5.20.6 SPYDialogAction

```
public class SPYDialogAction
{
    public final static long           = 0;
    spyDialogOK
    public final static long           = 1;
    spyDialogCancel
    public final static long           = 2;
    spyDialogUserInput
}
```


© 2007 Altova GmbH

5.20.7 SPYDOMType

```
public class SPYDOMType
{
   public final static long          = 0;
   spyDOMType msxml4
   public final static long          = 1;
   spyDOMType_xerces
}
```

5.20.8 SPYDTDSchemaFormat

```
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;
}
```

5.20.9 SPYEncodingByteOrder

```
public class SPYEncodingByteOrder
{
    public final static long spyNONE      = 0;
    public final static long              = 1;
    spyLITTLE ENDIAN
    public final static long              = 2;
    spyBIG ENDIAN
}
```

5.20.10 SPYExportNamespace

```
public class SPYExportNamespace
{
    public final static long spyNoNamespace                = 0;
    public final static long spyReplaceColonWithUnderscore = 1;
}
```

5.20.11 SPYFrequentElements

```
public class SPYFrequentElements
{
   public final static long         = 0;
   spyGlobalElements
   public final static long         = 1;
   spyGlobalComplexType
}
```

5.20.12 SPYLibType

```
public class SPYLibType
{
   public final static long          = 0;
   spyLibType static
   public final static long          = 1;
   spyLibType_dll
}
```

5.20.13 SPYLoading

```
public class SPYLoading
{
   public final static long          = 0;
   spyUseCacheProxy
   public final static long          = 1;
   spyReload
}
```

5.20.14 SPYNameDateTimeFormat

```
public class  SPYNumberDateTimeFormat
{
   public final static long           = 0;
   spySystemLocale
   public final static long           = 1;
   spySchemaCompatible
}
```

5.20.15 SPYProgrammingLanguage

```
public class SPYLoading
{
    public final static long spyUndefinedLanguage = -1;
    public final static long spyJava              = 0;
    public final static long spyCpp               = 1;
    public final static long spyCSharp            = 2;
}
```

5.20.16 SPYProjectItemTypes

```
public class SPYProjectItemTypes
{
   public final static long       = 0;
   spyUnknownItem
   public final static long       = 1;
   spyFileItem
   public final static long       = 2;
   spyFolderItem
   public final static long       = 3;
   spyURLItem
}
```


© 2007 Altova GmbH

5.20.17 SPYProjectType

```
public class SPYProjectType
{
   public final static long              = 0;
   spyVisualStudioProject
   public final static long              = 1;
   spyVisualStudio2003Project
   public final static long              = 2;
   spyBorlandProject
   public final static long              = 3;
   spyMonoMakefile
}
```

5.20.18 SPYSchemaDefKind

```
public class SPYSchemaDefKind
{
   public final static long spyKindElement  = 0;
   public final static long                 = 1;
   spyKindComplexType
   public final static long                 = 2;
   spyKindSimpleType
   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                 = 7;
   spyKindAttrGroup
   public final static long spyKindAttrAny  = 8;
   public final static long                 = 9;
   spyKindIdentityUnique
   public final static long                 = 10;
   spyKindIdentityKey
   public final static long                 = 11;
   spyKindIdentityKeyRef
   public final static long                 = 12;
   spyKindIdentitySelector
   public final static long                 = 13;
   spyKindIdentityField
   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;
}
```

5.20.19 SPYSchemaDocumentationFormat

```
public class SPYSchemaDocumentationFormat
{
   public final static long           = 0;
   spySchemaDoc HTML
```

```
    public final static long           = 1;
    spySchemaDoc MSWord
}
```

5.20.20 SPYTextDelimiters

```
public class SPYTextDelimiters
{
    public final static long           = 0;
    spyTabulator
    public final static long           = 1;
    spySemicolon
    public final static long spyComma  = 2;
    public final static long spySpace  = 3;
}
```

5.20.21 SPYTextEnclosing

```
public class SPYTextEnclosing
{
    public final static long       = 0;
    spyNoEnclosing
    public final static long       = 1;
    spySingleQuote
    public final static long       = 2;
    spyDoubleQuote
}
```

5.20.22 SPYTypeDetection

```
public class SPYTypeDetection
{
    public final static long          = 0;
    spyBestPossible
    public final static long          = 1;
    spyNumbersOnly
    public final static long          = 2;
    spyNoDetection
}
```

5.20.23 SPYURLTypes

```
public class SPYURLTypes
{
    public final static long        = (-1);
    spyURLTypeAuto
    public final static long        = 0;
    spyURLTypeXML
    public final static long        = 1;
    spyURLTypeDTD
}
```


© 2007 Altova GmbH

5.20.24 SpyViewModes

```
public class SPYViewModes
{
   public final static long          = 0;
   spyViewGrid
   public final static long          = 1;
   spyViewText
   public final static long          = 2;
   spyViewBrowser
   public final static long          = 3;
   spyViewSchema
   public final static long          = 4;
   spyViewContent
}
```

5.20.25 SPYWhitespaceComparison

```
public class SPYWhitespaceComparison
{
   public final static long            = 0;
   spyCompareAsIs
   public final static long            = 1;
   spyCompareNormalized
   public final static long            = 2;
   spyStripAll
}
```

5.20.26 SPYXMLDataKind

```
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             = 3;
   spyXMLDataXML
   public final static long             = 4;
   spyXMLDataElement
   public final static long             = 5;
   spyXMLDataAttr
   public final static long             = 6;
   spyXMLDataText
   public final static long             = 7;
   spyXMLDataCData
   public final static long             = 8;
   spyXMLDataComment
   public final static long             = 9;
   spyXMLDataPI
   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
}
```


© 2007 Altova GmbH

6 XMLSpy Integration

XMLSpyControl is a control that provides a means of integration of the XMLSpy user interface and the functionality described in this section into most kinds of applications. ActiveX technology was chosen so as to allow integration using any of a wide variety of languages; this enables C++, C#, VisualBasic, or HTML to be used for integration. All components are full OLE Controls, which makes integration as simple as possible. Two different levels of integration are provided, thus enabling the integration to be adapted to a wide range of needs.

For a successful integration you have to consider the following main design factors:

- What technology or programming language can the hosting application use to integrate the XMLSpyControl?
- Should the integrated UI look exactly like XMLSpy with all its menus, toolbars, and windows, or will a subset of these elements—like allowing only one document and a restricted set of commands—be more effective?
- How deep will the integration be? Should the XMLSpy user interface be used as is? Are user interface extensions and/or restrictions required? Can some frequently used tasks be automated?

The sections, Integration at the Application Level and Integration at Document Level, both of which have examples in various programming languages, will help you to make the right decisions quickly. The section, Object Reference, describes all COM objects that can be used for integration, together with their properties and methods.

For automation tasks, the XMLSpy Automation Interface is accessible from the XMLSpyControl as well.

6.1 Integration at the Application Level

Integration at application level is simple and straightforward. It allows you to embed the complete interface of XMLSpy into a window of your application. Since you get the whole user interface of XMLSpy, you get all menus, toolbars, the status bar, document windows, and helper windows. Customization of the application's user interface is restricted to what XMLSpy provides. This includes rearrangement and resizing of helper windows and customization of menus and toolbars.

The only ActiveX control you need to integrate is `XMLSpyControl`. Its property `IntegrationLevel` defaults to application-level. You may use `Appearance` and `BorderStyle` to configure the appearance of the control's wrapper window. Do not instantiate or access `XMLSpyControlDocument` or `XMLSpyControlPlaceHolder` ActiveX controls when integrating at application-level.

If you have any initialization to do or if you want to automate some behaviour of XMLSpy, use the properties, methods, and events described for `XMLSpyControl`. Consider using `XMLSpyControl.Application` for more complex access to XMLSpy functionality.

In this section is an example (Example: HTML) showing how the XMLSpy application can be embedded in an HTML page. For usage with other programming languages, or more sophisticated access, see the Examples of integration at document-level.

6.1.1 Example: HTML

This example shows a simple integration of the XMLSpy control at application-level into a HTML page. The integration is described in the following sections:

- Instantiate a XMLSpyControl in HTML code.
- Implement buttons to load documents and automate code-generation tasks.
- Define actions for some application events.

The code for this example is available at the following location in your XMLSpy installation: `Examples\ActiveX\HTML\XMLSpyActiveX_ApplicationLevel.htm`.

Instantiate the Control

The HTML `Object` tag is used to create an instance of the XMLSpyControl. The `Classid` is that of XMLSpyControl. Width and height specify the window size. No additional parameters are necessary, since application-level is the default.

```
<OBJECT id="objXMLSpyControl"
        Classid="clsid:a258bba2-3835-4c16-8590-72b44f52c471"
        width="1000"
        height="700"
        VIEWASTEXT>
</OBJECT>
```

Add Button to Open Default Document

As a simple example of how to automate some tasks, we add a button to the page:

```
<input type="button" value="Open Marketing Expenses"
onclick="BtnOpenMEFile()">
```

 © 2007 Altova GmbH

When clicked, a predefined document will be opened in the XMLSpyControl. We use a method to locate the file relative to the XMLSpyControl so the example can run on different installations.

```
<SCRIPT ID=Javahandlers LANGUAGE=javascript>
// --------------------------------
// open a pre-defined document
function BtnOpenMEFile()
{
 var pos = objXMLSpyControl.BaseHref.indexOf("ActiveX");

 if(pos > 7)
 {
  path = objXMLSpyControl.BaseHref.substr(7, pos - 7);            // remove
file protocol

  objXMLSpyControl.Open(path + "OrgChart.xml ");
 }
 else
 {
  alert("Unable to locate OrgChart.xml  at: " + objXMLSpyControl.BaseHref);
 }
}
</SCRIPT>
```

Add Buttons for Code Generation

For direct access, we want to have a button that will validate the current document. The method is similar to that used in the previous section.

First comes the button:

```
<input type="button" value="Validate" onclick="BtnValidate()">
```

Then we provide the script that will validate the current document.

```
<SCRIPT ID=Javahandlers LANGUAGE=javascript>
// ---------------------------------------------------------------------
// check validity of current document.
// if validation fails, show validation result in alert box .
function BtnValidate()
{
	// get top-level object of automation interface
	var objApp = objXMLSpyControl.Application;

	// get the active document
	var objDocument = objApp.ActiveDocument;

	if ( objDocument == null )
		alert( "no active document found" );
	else
	{
		// define as arrays to support their usage as return parameters
		var errorText = new Array(1);
		var errorPos = new Array(1);
		var badData = new Array(1);

		var valid = objDocument.IsValid(errorText, errorPos, badData);

		if (! valid)
		{
			// compose the error description
			var text = errorText;
```

```
                        // access that XMLData object only if filled in
                        if ( badData[ 0] != null)
                              text += "(" + badData[ 0]. Name + "/" +
badData[ 0]. TextValue + ")";


                        alert( "Validation error[" + errorPos + "]: " + text);
            }
            else
                  alert( "Docuent is valid"); }
}
</SCRIPT>
```

Connect to Custom Events

The example implements two event callbacks for XMLSpyControl custom events to show the principle:

```
<! -- ------------------------------------------------------------ -->
<! --  custom event ' OnDocumentOpened" of XMLSpyControl object  -->
<SCRIPT FOR="objXMLSpyControl" event="OnDocumentOpened( objDocument )"
LANGUAGE="javascript">
      // alert( "Document '" + objDocument. Name + "' opened! ");
</SCRIPT>

<! -- ------------------------------------------------------------ -->
<! --  custom event ' OnDocumentClosed" of XMLSpyControl object  -->
<SCRIPT FOR="objXMLSpyControl" event="OnDocumentClosed( objDocument )"
LANGUAGE="javascript">
      // alert( "Document '" + objDocument. Name + "' closed! ");
</SCRIPT>
```


© 2007 Altova GmbH

6.2 Integration at Document Level

Integration at document level gives you freedom over instantiation and placement of the following parts of the XMLSpy user interface:

- Editing windows for XMLSpy
- XMLSpy entry helper windows
- XMLSpy validator output window
- XMLSpy project window
- XMLSpy XPath profiler window
- XMLSpy XPath dialog window
- XMLSpy XSLT/XQuery debugger windows
- XMLSpy SOAP debugger window

If necessary, a replacement for the menus and toolbars of XMLSpy must be provided by your application.

You will need to instantiate and access multiple ActiveX controls, depending on which user interface parts you want to re-use. All these controls are contained in the XMLSpyControl OCX.

- Use XMLSpyControl to set the integration level and access application wide functionality.
- Use XMLSpyControlDocument to create any number of editor windows. It may be sufficient to create only one window and re-use it, depending on your needs.
- Optionally Use XMLSpyControlPlaceholder to embed XMLSpy entry helper windows, validator output or other windows mentioned above.
- Access run-time information about commands, menus, and toolbars available in XMLSpyControl to seamlessly integrate these commands into your application's menus and toolbars. See Use Commands for more information.

If you want to automate some behaviour of XMLSpy use the properties, methods, and events described for the XMLSpyControl, XMLSpyControlDocument and XMLSpyControlPlaceHolder. Consider using XMLSpyControl.Application, XMLSpyControlDocument.Document and XMLSpyControlPlaceHolder.Project for more complex access to XMLSpy functionality. However, to open a document always use XMLSpyControlDocument.Open or XMLSpyControlDocument.New on the appropriate document control. To open a project always use XMLSpyControlPlaceHolder.OpenProject on a placeholder control embedding a XMLSpy project window.

See Examples on how to instantiate and access the necessary controls in different programming environments.

6.2.1 Use XMLSpyControl

To integrate at document level, instantiate a `XMLSpyControl` first. Set the property `IntegrationLevel` to `IActiveXIntegrationOnDocumentLevel(=1)`. Set the window size of the embedding window to `0x0` to hide any user interface behind the control. You may use `Appearance` and `BorderStyle` to configure the appearance of the control's wrapper window.

Avoid using the method `Open` since this might lead to unexpected results. Use the corresponding open methods of `XMLSpyControlDocument` and `XMLSpyControlPlaceHolder`, instead.

See Query XMLSpy Commands for a description of how to integrate XMLSpy commands into your application. Send commands to XMLSpy via the method `Exec`. Query if a command is currently enabled or disabled using the method `QueryStatus`.

6.2.2 Use XMLSpyControlDocument

An instance of the `XMLSpyControlDocument` ActiveX control allows you to embed one XMLSpy document editing window into your application. You can use any number of instances you need.

Use the method `Open` to load any other existing file.

The control does not supports a read-only mode. The value of the property `ReadOnly` is ignored.

Use `Path` and `Save` or methods and properties accessible via the property `Document` to access document functionality.

6.2.3 Use XMLSpyControlPlaceHolder

Instances of XMLSpyControlPlaceHolder ActiveX controls allow you to selectively embed the additional helper windows of XMLSpy into your application. The property `PlaceholderWindowID` selects the XMLSpy helper window to be embedded. Use only one XMLSpyControlPlaceHolder for each window identifier. See `Enumerations.XMLSpyControlPlaceholderWindow` for valid window identifiers.

For placeholder controls that select the XMLSpy project window, additional methods are available. Use `OpenProject` to load a XMLSpy project. Use the property `Project` and the methods and properties from the XMLSpy automation interface to perform any other project related operations.

6.2.4 Query XMLSpy Commands

When integrating at document-level, no menu or toolbar from XMLSpy is available to your application. Instead, you can query all the commands and the structure of the application menu at runtime. Use the property `XMLSpyControl.CommandsStructure` to access this information. Professional applications will need to integrate this menu in a sophisticated manner into their own menu structure. Your installation of XMLSpy even provides you with command label images used within XMLSpy. See the folder `Examples\ActiveX\Images` of your XMLSpy installation for icons in GIF format. The file names correspond to the `labels` of commands.

6.2.5 Examples

This section contains examples of XMLSpy document-level integration using different container environments and programming languages. Source code for all examples is available in the folder `Examples\ActiveX` of your XMLSpy installation.

C#

The C# example shows how to integrate the XMLSpyControl in a common desktop application created with C# using Visual Studio .NET 2002. The following topics are covered:

- Integration of a XMLSpyControl Document control to embed a XMLSpy document editing window.
- Usage of a XMLSpyControlPlaceholder controls to embed the XMLSpy XPath dialog

 © 2007 Altova GmbH

window.
- Opening and closing of XMLSpy documents via the main menu.

Please note that the example application is already complete. There is no need to change anything if you want to run and see it working. The following steps describe what general actions and considerations must be taken in order to create a project such as this.

Introduction

Adding the XMLSpy components to the Toolbox
Before you take a look at the sample project please add the assemblies to the .NET IDE Toolbox. The XMLSpy Installer will have already installed the assemblies in the .NET Global Assembly Cache (GAC). If you open the Toolbox dialog under **Tools | Add/Remove Toolbox Items** the controls will appear as `AxXMLSpyControl`, `AxXMLSpyControlDocument` and `AxXMLSpyControlPlaceholder` on the .NET Framework Components tab. Check all to make them available to the IDE.

Now you can open the `XPathDialog.sln` file in the `ActiveX\C#\XPathDialog` folder to load the project.

Placing the XMLSpyControl

It is necessary to have one XMLSpyControl instance to set the integration level and to manage the Document and Placeholder controls of the XMLSpy library. The control is accessible via the General section of the Toolbox helper window in the IDE. To add it you need to select the component in the Toolbox window and drag a rectangle wherever you want to have it in the destination window. If you have an application which does not open a window on startup you can use a simple invisible Form with the control on it which is created manually in the code.

The example project adds this instance to the main MdiContainer MDIMain. If you open MDIMain in the Design View from the Solution Explorer you will see a light blue rectangle at the top-left side in the client area of the Frame window. Selecting this rectangle will show you the properties of the XMLSpyControl. It is important to set the `IntegrationLevel` property to **`ICActiveXIntegrationOnDocumentLevel`** in order to turn on the Document and Placeholder support of the XMLSpy library. You can set the Visible flag to False to avoid any confusion about the control for the user.

Adding the Document Control

Document editing window on the MDI Frame
The example project uses one Document control show and edit one XMLSpy document in the main MDI Frame. It is added via the Toolbox window by dragging a rectangle on the destination Form.The Document control also has the Anchor and Dock properties set in order to react on resizing of the Frame window.

Adding the Placeholder Control

Placeholders on the MDI Frame
The example project uses one Placeholder control to embed the XPath Dialog window into the main MDI Frame. It is added via the Toolbox window by dragging a rectangle on the destination Form. To set the type of the Placeholder which should be displayed one has to set the `PlaceholderWindowID` property. This property can also be changed during runtime in the

code of the application. The Placeholder control would change its content immediately. The Placeholder also has the Anchor properties set in order to react on resizing of the Frame window.

Retrieving Command Information

The XMLSpyControl gives access to all commands of XMLSpy through its CommandsStructure property. The example project does not use this information but it is easy to use the Commands and Command interfaces to dynamically build a menu in the MDI Frame window.

The code to add the commands would be placed into the MDIMain method of the XMLSpyApplication class in the file MDIMain.cs:

```
public MDIMain()
{
    .
    .
    .
    XMLSpyLib.Commands objCommands;
    objCommands = axXMLSpyControl.CommandsStructure;

    long nCount = objCommands.Count;

    for(long idx = 0;idx < nCount;idx++)
    {
        XMLSpyLib.Command objCommand;
        objCommand = objCommands[(int)idx];

        // We are looking for the Menu with the name IDR_XMLSPY. This menu
        contains
        // the complete main menu of XMLSpy.

        if(objCommand.Label == "IDR_XMLSPY")
        {
            InsertMenuStructure(mainMenu.MenuItems, 1, objCommand, 0, 0,
            false);
        }
    }
    .
    .
    .
}
```

mainMenu is the name of the menu object of the MDI Frame window created in the Visual Studio IDE. InsertMenuStructure takes the XMLSpy menu from the IDR_XMLSPY command object and adds the XMLSpy menu structure to the already existing menu of the sample project. No commands from the **File**, **Project**, or **Window** menu are added.

The new commands are instances of the class CustomMenuItem, which is defined in CustomMenuItem.cs. This class has an additional member to save the XMLSpy command ID, which is taken to execute the command using Exec on selecting the menu item. This code from InsertMenuStructure creates the new command:

```
CustomMenuItem  newMenuItem = new CustomMenuItem();

if(objCommand.IsSeparator)
    newMenuItem.Text = "-";
else
{
    newMenuItem.Text = strLabel;
```

 © 2007 Altova GmbH

```
    newMenuItem.m_XMLSpyCmdID = (int)objCommand.ID;
    newMenuItem.Click += new EventHandler(AltovaMenuItem_Click);
}
```

You can see that all commands get the same event handler AltovaMenuItem_Click which does the processing of the command:

```
private void AltovaMenuItem_Click(object sender, EventArgs e)
{
    if(sender.GetType() ==
    System.Type.GetType("XMLSpyApplication.CustomMenuItem"))
    {
        CustomMenuItemcustomItem = (CustomMenuItem)sender;

        ProcessCommand(customItem.m_XMLSpyCmdID);
    }
}

private void ProcessCommand(int nID)
{
    XMLSpyDoc docXMLSpy = GetCurrentXMLSpyDoc();

    if(docXMLSpy != null)
        docXMLSpy.axXMLSpyControlDoc.Exec(nID);
    else
        axXMLSpyControl.Exec(nID);
}
```

`ProcessCommand` delegates the execution either to the XMLSpyControl itself or to any active XMLSpy document loaded in a `XMLSpyControlDocument` control. This is necessary because the XMLSpyControl has no way to know which document is currently active in the hosting application.

Testing the Example

After adding the assemblies to the Toolbox (see Introduction), you can run the sample project with F5 without the need to change anything in the code. The main MDI Frame window is created and contains an editing window with an empty XML document and a XPath Dialog window of XMLSpy at the bottom. Use **File | Open** to open the file `OrgChart.xml`, which is in the XMLSpy examples folder. The file is loaded and displayed.

After you load the document, you can try using the XPath dialog.

HTML

This example shows an integration of the XMLSpy control at document-level into a HTML page. The following topics are covered:

- Instantiate a XMLSpyControl ActiveX control object in HTML code
- Instantiate a XMLSpyControlDocument ActiveX control to allow editing a XMLSpy file
- Instantiate one XMLSpyControlPlaceHolder to alternatively host one of the XMLSpy entry helper windows
- Instantiate one XMLSpyControlPlaceHolder ActiveX control to show the XMLSpy validation output window
- Create a simple customer toolbar for some heavy-used XMLSpy commands
- Add some more buttons that use the COM automation interface of XMLSpy

- Use event handlers to update command buttons

This example is available in its entirety in the file `XMLSpyActiveX_ApplicationLevel.htm` within the `Examples\ActiveX\HTML\` folder of your XMLSpy installation.

Instantiate the XMLSpyControl

The HTML `OBJECT` tag is used to create an instance of the XMLSpyControl. The Classid is that of XMLSpyControl. Width and height are set to 0 since we use this control as manager control without use for its user interface. The integration level is specified as a parameter within the `OBJECT` tag.

```
<OBJECT id="objXMLSpyX"
        Classid="clsid: a258bba2-3835-4c16-8590-72b44f52c471"
        width="0"
        height="0"
        VIEWASTEXT>
   <PARAM NAME="IntegrationLevel" VALUE="1">
</OBJECT>
```

Create Editor window

The HTML `OBJECT` tag is used to embed an editing window for XMLSpy documents.

```
<OBJECT id="objDoc1"
        Classid="clsid: DFBB0871-DAFE-4502-BB66-08CEB7DF5255"
        width="600"
        height="500"
        VIEWASTEXT>
</OBJECT>
```

Create Project Window

The HTML `OBJECT` tag is used to create a XMLSpyControlPlaceHolder window. The first additional custom parameter defines the placeholder to show the XMLSpy project window. The second parameter loads one of the example projects delivered coming with your XMLSpy installation.

```
<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.mfp">
</OBJECT>
```

Create Placeholder for XMLSpy Helper Windows

The HTML `OBJECT` tag is used to instantiate a XMLSpyControlPlaceHolder ActiveX control that can host the different XMLSpy helper window. Initially, no helper window is shown.

```
<OBJECT id=" objEHWindow"
        Classid="clsid: 135DEEF4-6DF0-47c2-8F8C-F145F5F3F672"
```

 © 2007 Altova GmbH

```
        width="200"
        height="200"
        VIEWASTEXT>
   <PARAM name="PlaceholderWindowID" value="0">
</OBJECT>
```

Three buttons allow us to switch the actual window that will be shown. The JavaScript execute on-button-click sets the property `PlaceHolderWindowID` to the corresponding value defined in `XMLSpyControlPlaceholderWindow`.

```
<input type="button" value="EH - Elements" onclick="BtnHelperWindow(0)">
<input type="button" value="EH - Attributes" onclick="BtnHelperWindow(1)">
<input type="button" value="EH - Entities" onclick="BtnHelperWindow(2)">

<SCRIPT ID="Javahandlers" LANGUAGE="javascript">
//
---------------------------------------------------------------------------
// specify which of the helper windows shall be shown in the placeholder
control.
function BtnHelperWindow(i_ePlaceholderWindowID)
{
       objEHWindow.PlaceholderWindowID = i_ePlaceholderWindowID;
}
</SCRIPT>
```

Create a Custom Toolbar

The custom toolbar consists of buttons with images of XMLSpy commands. The command ID numbers can be found in `Command Table`

```
.
<button id="btnWellFormed" title="Check Well-formedness"
onclick="BtnDoCommand(34049)">
       <img src="..\Images\IDC_CHECK_WELL_FORM.gif" width="16" height="16" />
</button>
<button id="btnValidate" title="Validate" onclick="BtnDoCommand(34174)">
       <img src="..\Images\IDC_VALIDATE.gif" width="16" height="16" />
</button>
```

On clicking one of these buttons the corresponding command Id is sent to the manager control.

```
<SCRIPT ID="Javahandlers" LANGUAGE="javascript">
// ------------------------------------------------------------
// perform any command specified by cmdID.
// command routing includes application, active document and view.
function BtnDoCommand(cmdID)
{
       objXMLSpyX.Exec( cmdID );
       msgtext.innerText = "Command " + cmdID + " performed.";
}
</SCRIPT>
```

Create More Buttons

In the example, we add some more buttons to show some automation code.

```
<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')">
```

The corresponding JavaScript looks like this:

```
<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:

	var pos = objXMLSpyX.BaseHref.indexOf("ActiveX");

	if(pos > 7)
	{
		path = objXMLSpyX.BaseHref.substr(7, pos - 7);		// remove
file protocol

		objDocCtrl.Open(path + strFileName);
		objDocCtrl.setActive();
	}
	else
	{
		alert("Unable to locate " + strFileName + " at: " +
objXMLSpyX.BaseHref);
	}
}

// ------------------------------------------------------------------------
// 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.
```

 © 2007 Altova GmbH

```
// if validation fails, show validation result in alert box .
function BtnValidate()
{
	// get top-level object of automation interface
	var objApp = objXMLSpyControl.Application;

	// get the active document
	var objDocument = objApp.ActiveDocument;

	if ( objDocument == null )
		alert( "no active document found" );
	else
	{
		// define as arrays to support their usage as return parameters
		var errorText = new Array(1);
		var errorPos = new Array(1);
		var badData = new Array(1);

		var valid = objDocument.IsValid(errorText, errorPos, badData);

		if (! valid)
		{
			// compose the error description
			var text = errorText;

			// access that XMLData object only if filled in
			if (badData[0] != null)
				text += "(" + badData[0].Name + "/" +
badData[0].TextValue + ")";


			alert("Validation error[" + errorPos + "]: " + text);
		}
		else
			alert("Docuent is valid");
	}
}
</SCRIPT>
```

Create Event Handler to Update Button Status

Availability of a command may vary with every mouseclick or keystroke. The custom event `OnUpdateCmdUI` of XMLSpyControl gives us an opportunity to update the enabled/disabled state of buttons associated with XMLSpy commands. The method `XMLSpyControl.QueryStatus` is used to query whether a command is enabled or not.

```
<SCRIPT FOR="objXMLSpyX" event="OnUpdateCmdUI()" LANGUAGE="javascript">
	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
	}
</SCRIPT>
```

6.3 Command Table

Tables in this section list the names and identifiers of all commands that are available within XMLSpy. Every sub-section lists the commands from the corresponding top-level menu of XMLSpy. The left-most column shows the command's menu text to make it easier for you to identify the functionality behind the command. The last sub-section is a collection of those commands that are not accessible via the main menu.

Depending on the edition of XMLSpy you have installed, some of these commands might not be supported. See Query XMLSpy Commands on how to query the current resource structure and command availability. The same topics shows how to use the same command icons that are used by XMLSpy if you are not already integrating on application-level.

Use the command identifiers with XMLSpyControl.QueryStatus or XMLSpyControlDocument.QueryStatus to check the current status of a command. Use XMLSpyControl.Exec or XMLSpyControlDocument.Exec to execute a command.

File Menu
Edit Menu
Project Menu
XML menu
DTD/Schema Menu
Schema Design Menu
XSL/XQuery Menu
Authentic Menu
Convert Menu
View Menu
Browser Menu
WSDL Menu
SOAPMenu
Tools Menu
Window Menu
Help Menu

6.3.1 File Menu

Commands from the File menu:

Menu Text	Command Name	ID
New...	ID_FILE_NEW	57600
Open...	ID_FILE_OPEN	57601
Open URL...	IDC_OPEN_URL	34097
Reload	IDC_FILE_RELOAD	34065
Encoding...	IDC_ENCODING	34061
Close	ID_FILE_CLOSE	57602
Close All	IDC_CLOSE_ALL	34050
Save	ID_FILE_SAVE	57603
Save As...	ID_FILE_SAVE_AS	57604
Save to URL...	ID_FILE_SAVE_TO_URL	34209
Save All	ID_FILE_SAVE_ALL	34208
Send by Mail...	ID_FILE_SEND_MAIL	57612

© 2007 Altova GmbH

Print...	IDC_PRINT	34103
Print Preview	IDC_PRINT_PREVIEW	34104
Print Setup...	ID_FILE_PRINT_SETUP	57606
Recent File	ID_FILE_MRU_FILE1	57616
Exit	ID_APP_EXIT	57665

6.3.2 Edit Menu

Commands from the Edit menu:

Menu Text	Command Name	ID
Undo	ID_EDIT_UNDO	57643
Redo	ID_EDIT_REDO	57644
Cut	ID_EDIT_CUT	57635
Copy	ID_EDIT_COPY	57634
Paste	ID_EDIT_PASTE	57637
Delete	ID_EDIT_CLEAR	57632
Copy as XML-Text	IDC_COPY_AS_XML_TEXT	33443
Copy as Structured text	IDC_COPY_AS_STRUCTURED_TEXT	33442
Copy XPath	IDC_COPY_XPATH	33444
Pretty-Print XML Text	IDC_PRETTY_PRINT	34101
Select All	ID_EDIT_SELECT_ALL	57642
Find...	ID_EDIT_FIND	57636
Find next	ID_EDIT_REPEAT	57640
Replace...	ID_EDIT_REPLACE	57641
Find in files...	IDC_FIND_IN_FILES	34000
Insert/Remove Bookmark	IDC_TOGGLE_BOOKMARK	34162
Remove All Bookmarks	IDC_REMOVEALLBOOKMARKS	34132
Goto Next Bookmark	IDC_GOTONEXTBOOKMARK	34070
Goto Previous Bookmark	IDC_GOTOPREVBOOKMARK	34071

6.3.3 Project Menu

Commands from the Project menu:

Menu Text	Command Name	ID
New Project	IDC_PROJECT_NEW	34122
Open Project...	IDC_PROJECT_OPEN	34123
Reload Project	IDC_PROJECT_RELOAD	34126
Close Project	IDC_PROJECT_CLOSE	34113

Save Project	IDC_PROJECT_SAVE	34127
Source Control/Open Project...	IDC_SCC_OPEN_PROJECT	34140
Source Control/Enable Source Code Control	IDC_SCC_ENABLE	34137
Source Control/Get latest version...	IDC_SCC_GET	34138
Source Control/Check Out...	IDC_SCC_CHECK_OUT	34135
Source Control/Check In...	IDC_SCC_CHECK_IN	34134
Source Control/Undo Check Out...	IDC_SCC_UNDO_CHECK_OUT	34145
Source Control/Add to Source Control...	IDC_SCC_ADD	34133
Source Control/Remove from Source Control...	IDC_SCC_REMOVE	34143
Source Control/Show History...	IDC_SCC_HISTORY	34139
Source Control/Show Differences...	IDC_SCC_DIFF	34136
Source Control/Properties...	IDC_SCC_PROPERTIES	34141
Source Control/Refresh Status...	IDC_SCC_REFRESH	34142
Source Control/Run Native Interface...	IDC_SCC_RUN	34144
Add Files to Project...	IDC_PROJECT_ADD_FILES	34109
Add URL to Project...	IDC_PROJECT_ADD_URL	34112
Add Active File to Project	IDC_PROJECT_ADD_ACTIVE	34106
Add Active and Related Files to Project	IDC_PROJECT_ADD_RELATED	34111
Add Project Folder to Project...	IDC_PROJECT_ADD_FOLDER	34110
Add External Folder to Project...	IDC_PROJECT_ADD_EXT_FOLDER	34107
Add External Web Folder to Project...	IDC_PROJECT_ADD_EXT_URL_FOLDER	34108
Project Properties...	IDC_PROJECT_PROPERTIES	34124
Recent Project	IDC_RECENT_PROJECT_ID	34265

6.3.4 XML menu

Commands from the XML menu:

Menu Text	Command Name	ID
Insert/Attribute	IDC_INSERT_ATTRIBUTE	33449
Insert/Element	IDC_INSERT_STRUCT	33459
Insert/Text	IDC_INSERT_TEXT	33460
Insert/CData	IDC_INSERT_CDATA	33450
Insert/Comment	IDC_INSERT_COMMENT	33451
Insert/XML	IDC_INSERT_XML	33461
Insert/Processing Instruction	IDC_INSERT_PI	33458
Insert/DOCTYPE	IDC_INSERT_DEF_DOCTYPE	33453
Insert/ExternalID	IDC_INSERT_DEF_EXTERNAL_ID	33456
Insert/ELEMENT	IDC_INSERT_DEF_ELEMENT	33454
Insert/ATTLIST	IDC_INSERT_DEF_ATTLIST	33452
Insert/ENTITY	IDC_INSERT_DEF_ENTITY	33455

© 2007 Altova GmbH

Insert/NOTATION	IDC_INSERT_DEF_NOTATION	33457
Append/Attribute	IDC_APPEND_ATTRIBUTE	33415
Append/Element	IDC_APPEND_STRUCT	33425
Append/Text	IDC_APPEND_TEXT	33426
Append/CData	IDC_APPEND_CDATA	33416
Append/Comment	IDC_APPEND_COMMENT	33417
Append/XML	IDC_APPEND_XML	33427
Append/Processing Instruction	IDC_APPEND_PI	33424
Append/DOCTYPE	IDC_APPEND_DEF_DOCTYPE	33419
Append/ExternalID	IDC_APPEND_DEF_EXTERNAL_ID	33422
Append/ELEMENT	IDC_APPEND_DEF_ELEMENT	33420
Append/ATTLIST	IDC_APPEND_DEF_ATTLIST	33418
Append/ENTITY	IDC_APPEND_DEF_ENTITY	33421
Append/NOTATION	IDC_APPEND_DEF_NOTATION	33423
Add child/Attribute	IDC_ADD_CHILD_ATTRIBUTE	33402
Add child/Element	IDC_ADD_CHILD_STRUCT	33412
Add child/Text	IDC_ADD_CHILD_TEXT	33413
Add child/CData	IDC_ADD_CHILD_CDATA	33403
Add child/Comment	IDC_ADD_CHILD_COMMENT	33404
Add child/XML	IDC_ADD_CHILD_XML	33414
Add child/Processing Instruction	IDC_ADD_CHILD_PI	33411
Add child/DOCTYPE	IDC_ADD_CHILD_DEF_DOCTYPE	33406
Add child/ExternalID	IDC_ADD_CHILD_DEF_EXTERNAL_ID	33409
Add child/ELEMENT	IDC_ADD_CHILD_DEF_ELEMENT	33407
Add child/ATTLIST	IDC_ADD_CHILD_DEF_ATTLIST	33405
Add child/ENTITY	IDC_ADD_CHILD_DEF_ENTITY	33408
Add child/NOTATION	IDC_ADD_CHILD_DEF_NOTATION	33410
Convert to/Attribute	IDC_CONVERT_TO_ATTRIBUTE	33429
Convert to/Element	IDC_CONVERT_TO_STRUCT	33439
Convert to/Text	IDC_CONVERT_TO_TEXT	33440
Convert to/CData	IDC_CONVERT_TO_CDATA	33430
Convert to/Comment	IDC_CONVERT_TO_COMMENT	33431
Convert to/XML	IDC_CONVERT_TO_XML	33441
Convert to/Processing Instruction	IDC_CONVERT_TO_PI	33438
Convert to/DOCTYPE	IDC_CONVERT_TO_DEF_DOCTYPE	33433
Convert to/ExternalID	IDC_CONVERT_TO_DEF_EXTERNAL_ID	33436
Convert to/ELEMENT	IDC_CONVERT_TO_DEF_ELEMENT	33434
Convert to/ATTLIST	IDC_CONVERT_TO_DEF_ATTLIST	33432
Convert to/ENTITY	IDC_CONVERT_TO_DEF_ENTITY	33435
Convert to/NOTATION	IDC_CONVERT_TO_DEF_NOTATION	33437

Table/Display as Table	IDC_GRID_VIEW_AS_TABLE	34075
Table/Insert Row	IDC_TABLE_INSERT_ROW	34158
Table/Append Row	IDC_TABLE_APPEND_ROW	34157
Table/Ascending Sort	IDC_TABLE_SORT_ASC	33464
Table/Descending Sort	IDC_TABLE_SORT_DESC	33465
Move left	IDC_MOVE_LEFT	34091
Move right	IDC_MOVE_RIGHT	34092
Enclose in Element	IDC_ENCLOSE_IN_ELEMENT	33446
Evaluate XPath...	IDC_EVALUATE_XPATH	34007
Check well-formedness	IDC_CHECK_WELL_FORM	34049
Validate	IDC_VALIDATE	34174
Update Entry Helpers	IDC_UPDATE_ELEMENT_CHOICE	34173
Namespace prefix...	IDC_NAMESPACE	33462

6.3.5 DTD/Schema Menu

Commands from the DTD/Schema menu:

Menu Text	Command Name	ID
Assign DTD...	IDC_ASSIGN_DTD	34032
Assign Schema...	IDC_ASSIGN_SCHEMA	34033
Include another DTD...	IDC_INCLUDE_DTD	34084
Go to DTD	IDC_GOTO_DTD	34072
Go to Schema	IDC_GOTO_SCHEMA	34074
Go to Definition	IDC_GOTO_DEFINITION	33447
Generate DTD/Schema...	IDC_GENERATE_DTD_SCHEMA	34068
Convert DTD/Schema...	IDC_CONVERT_DTD_SCHEMA	34052
Convert to UML...	IDC_CONVERT_SCHEMA_TO_UML	34008
Map to other DTD/Schema or DB in MapForce...	IDC_DTD_OPENIN_MAPFORCE	34056
Design HTML/PDF Output in StyleVision...	IDC_DTD_OPENIN_STYLEVISION	34057
Generate Sample XML File...	IDC_GENERATE_XML_FROM_SCHEMA	34069
Generate Program Code...	IDC_GENERATE_CODE_FROM_SCHEMA	34067
Flush memory cache	IDC_FLUSH_CACHED_FILES	34066

6.3.6 Schema Design Menu

Commands from the Schema Design menu:

Menu Text	Command Name	ID
Schema Settings	IDC_SCHEMA_NAMESPACES	33571
Save Diagram...	IDC_SCHEMA_SAVE_DIAGRAM	33581

© 2007 Altova GmbH

Generate Documentation	IDC_SCHEMA_DOCUMENTATION	34146
Configure view	IDC_SCHEMA_VIEW_CONFIG	33593
Zoom	IDC_SCHEMA_ZOOM	34150
Display All Globals	IDC_SCHEMA_MODE_GLOBALS	34147
Display Diagram	IDC_SCHEMA_MODE_DIAGRAM	33570
Enable Oracle Schema Extensions	IDC_SCHEMA_ORACLE_EXTENSIONS	33577
Oracle Schema Settings...	IDC_SCHEMA_ORACLE_SCHEMA_SETTINGS	33578
Enable Microsoft SQL Server Schema Extensions	IDC_SCHEMA_SQLSERVER_EXTENSIONS	33588
Named Schema Relationships...	IDC_SCHEMA_SQLSERVER_GLOBAL_RELATIONSHIPS	33589
Unnamed Element Relationships...	IDC_SCHEMA_SQLSERVER_LOCAL_RELATIONSHIPS	33590
Enable Tamino Schema Extensions	IDC_SCHEMA_TAMINO_EXTENSIONS	33591
Tamino Schema Properties...	IDC_SCHEMA_TAMINO_GLOBAL_SETTINGS	34149
Connect to SchemaAgent Server...	IDC_SCHEMA_SCHEMAAGENT_SERVER_CONNECT	33582
Disconnect from SchemaAgent Server	IDC_SCHEMA_SCHEMAAGENT_SERVER_DISCONNECT	33583
Show in SchemaAgent/Schema only	IDC_SCHEMAAGENT_SHOW	33504
Show in SchemaAgent/Schema and all directly referenced	IDC_SCHEMAAGENT_SHOW_RELATED	33507
Show in SchemaAgent/Schema and all referenced	IDC_SCHEMAAGENT_SHOW_ALL	33505
Show in SchemaAgent/Schema and all linked	IDC_SCHEMAAGENT_SHOW_ALL_REACHABLE	33506
Extended Validation	IDC_SCHEMA_EXTVALID_MENU	33539

6.3.7 XSL/XQuery Menu

Commands from the XSL/XQuery menu:

Menu Text	Command Name	ID
XSL Transformation	IDC_TRANSFORM_XSL	33006
XSL:FO Transformation	IDC_TRANSFORM_XSLFO	33007
XSL Parameters/XQuery Variables...	IDC_TRANSFORM_XSL_PARAMS	33008
XQuery Execution	IDC_TRANSFORM_XQUERY	34170
Enable XSLT 2 / XQuery profiling....	IDC_PROFILING_OPTIONS	34105

Assign XSL...	IDC_ASSIGN_XSL	33001
Assign XSL:FO...	IDC_ASSIGN_XSLFO	33002
Assign sample XML file...	IDC_ASSIGN_SAMPLE_XML	33000
Go to XSL	IDC_GOTO_XSL	33004
Start Debugger/Go	ID_PROCESS_XSL	34212
Stop Debugger	ID_XSLT_DEBUGGER_STOP	33017
Restart Debugger	ID_XSLT_DEBUGGER_RESTART	33013
End Debugger Session	ID_XSLT_DEBUGGER_END_SESSION	33011
Step Into	ID_XSLT_DEBUGGER_STEP	33014
Step Out	ID_XSLT_DEBUGGER_STEP_OUT	33015
Step Over	ID_XSLT_DEBUGGER_STEP_OVER	33016
Show Current Execution Node	ID_XSLT_DEBUGGER_GO_TO_CURRENT_EXECUTION_NODES	33012
Insert/Remove Breakpoint	IDC_TOGGLE_BREAKPOINT	34246
Insert/Remove Tracepoint	IDC_TOGGLE_TRACEPOINT	34248
Enable/Disable Breakpoint	IDC_ENABLE_BREAKPOINT	34245
Enable/Disable Tracepoint	IDC_ENABLE_TRACEPOINT	34247
Breakpoints/Tracepoints...	ID_XSLTDEBUGGER_BREAKPOINTS	33009
Debug Windows/Call Stack	ID_XSL_DEBUGWINDOWS_CALLSTACK	34238
Debug Windows/XPath-Watch	ID_XSL_DEBUGWINDOWS_WATCH	34244
Debug Windows/Context	ID_XSL_DEBUGWINDOWS_CONTEXT	34239
Debug Windows/Variables	ID_XSL_DEBUGWINDOWS_VARIABLE	34243
Debug Windows/Messages	ID_XSL_DEBUGWINDOWS_MESSAGES	34240
Debug Windows/Templates	ID_XSL_DEBUGWINDOWS_TEMPLATES	34241
Debug Windows/Info	ID_XSLXQUERY_DEBUGWINDOWS_INFO	34237
Debug Windows/Trace	ID_XSL_DEBUGWINDOWS_TRACES	34242
XSLT/XQuery Settings...	ID_XSLTDEBUGGER_SETTINGS	33010

© 2007 Altova GmbH

6.3.8 Authentic Menu

Commands from the Authentic menu:

Menu Text	Command Name	ID
New Document...	IDC_AUTHENTIC_NEW_FILE	34036
Edit Database Data...	IDC_AUTHENTIC_EDIT_DB	34035
Assign a StyleVision Stylesheet...	IDC_ASSIGN_SPS	34034
Edit StyleVision Stylesheet	IDC_EDIT_SPS	34060
Define XML Entities...	IDC_DEFINE_ENTITIES	32805
Hide markup	IDC_MARKUP_HIDE	34087
Show Small markup	IDC_MARKUP_SMALL	34090
Show Large markup	IDC_MARKUP_LARGE	34088
Show Mixed markup	IDC_MARKUP_MIX	34089
Append row	IDC_ROW_APPEND	32806
Insert row	IDC_ROW_INSERT	32809
Duplicate row	IDC_ROW_DUPLICATE	32808
Move row Up	IDC_ROW_MOVE_UP	32811
Move row Down	IDC_ROW_MOVE_DOWN	32810
Delete row	IDC_ROW_DELETE	32807

6.3.9 Convert Menu

Commands from the Convert menu:

Menu Text	Command Name	ID
Import Text file...	IDC_IMPORT_TEXT	34082
Import Database data...	IDC_IMPORT_DATABASE	34080
Import Microsoft Word document...	IDC_IMPORT_WORD	34083
Create XML Schema from DB Structure	IDC_CREATE_DB_SCHEMA	34054
DB Import based on XML Schema	IDC_IMPORT_DB_SCHEMA	34081
Create DB Structure from XML Schema	IDC_CREATE_DB_BASED_ON_SCHEMA	34053
Export to Text files...	IDC_EXPORT_TEXTFILE	34064
Export to a Database...	IDC_EXPORT_DB	34003
Tamino/Search ...	IDC_TAMINO_XQUERY	34161
Tamino/List Schema ...	ID_TAMINO_GETSCHEMA	34229
Tamino/Generate Sample XML File...	IDC_GENERATE_XML_FROM_SCHEMA	34069
Tamino/Add Document	IDC_TAMINO_ADDFILE	34159
Tamino/Delete Document	ID_TAMINO_DELETE_FILE	34228
Tamino/Get Active Document	IDC_TAMINO_GETFILE	34160
Tamino/Refresh	ID_TAMINO_REFRESH_DOC	34230
Tamino/Save to Tamino ...	IDC_SCHEMA_TAMINO_DEFINE	34148

Tamino/Tamino Schema Properties...	IDC_SCHEMA_TAMINO_GLOBAL_SETTINGS	34149
Oracle XML DB/Search...	ID_CONVERT_ORACLEXMLDB_QUERY	34207
Oracle XML DB/List Schemas...	ID_ORACLEXMLDB_LISTSCHEMAS	34211
Oracle XML DB/Add Schema...	ID_CONVERT_ORACLEXMLDB_ADDSCHEMA	34204
Oracle XML DB/Browse Oracle XML Documents...	ID_CONVERT_ORACLEXMLDB_BROWSE	34205
Oracle XML DB/Properties...	ID_CONVERT_ORACLEXMLDB_PROPERTIES	34206

6.3.10 View Menu

Commands from the View menu:

Menu Text	Command Name	ID
Text view	IDC_VIEW_TEXT	34180
Enhanced Grid view	IDC_VIEW_GRID	34178
Schema/WSDL Design view	IDC_VIEW_SCHEMA	34179
Authentic view	IDC_VIEW_CONTENT	34177
Browser view	IDC_VIEW_BROWSER	34176
Expand +	IDC_SEL_EXPAND	34152
Collapse -	IDC_SEL_COLLAPSE	34151
Expand fully	IDC_SEL_EXPAND_ALL	33463
Collapse unselected	IDC_COLLAPSE_UNSELECTED	33428
Optimal widths	IDC_OPTIMAL_WIDTHS	34099
Word Wrap	IDC_WORD_WRAP	34181
Go to line/char	IDC_GOTO_LINE	34073
Go to File	IDC_GOTO_FILE	33448
Line Numbers Margin	IDC_TOGGLE_NUMLINEMARGIN	34168
Bookmarks Margin	IDC_TOGGLE_BOOKMARKMARGIN	34163
Source Folding Margin	IDC_TOGGLE_FOLDINGMARGIN	34166
Indentation Guides	IDC_TOGGLE_INDENTGUIDES	34167

6.3.11 Browser Menu

Commands from the Browser menu:

Menu Text	Command Name	ID
Back	IDC_BROWSER_BACK	34039
Forward	IDC_BROWSER_FORWARD	34045
Stop	IDC_BROWSER_STOP	34047
Refresh	IDC_BROWSER_REFRESH	34046
Fonts/Largest	IDC_BROWSER_FONT_LARGEST	34041
Fonts/Larger	IDC_BROWSER_FONT_LARGE	34040

© 2007 Altova GmbH

Fonts/Medium	IDC_BROWSER_FONT_MEDIUM	34042
Fonts/Smaller	IDC_BROWSER_FONT_SMALL	34043
Fonts/Smallest	IDC_BROWSER_FONT_SMALLEST	34044
Separate window	IDC_BROWSER_USE_OWN_FRAME	34048

6.3.12 WSDL Menu

Commands from the WSDL menu:

Menu Text	Command Name	ID
Messages/Insert message	ID_WSDL_MESSAGES_ADDNEWMESSAGE	33715
Messages/Delete message	ID_WSDL_MESSAGES_DELETESELECTEDMESSAGE	33717
Messages/Add message part (parameter)	ID_WSDL_MESSAGES_ADDMESSAGEPART	33714
Messages/Delete message part (parameter)	ID_WSDL_MESSAGES_DELETEMESSAGEPART	33716
Operations/Insert Operation	ID_WSDL_OPERATIONS_INSERTOPERATION	33725
Operations/Delete operation	ID_WSDL_OPERATIONS_DELETEOPERATION	33724
Operations/Add input element	ID_WSDL_OPERATIONS_ADDINPUTFUNCTION	33719
Operations/Add output element	ID_WSDL_OPERATIONS_ADDOUTPUTFUNCTION	33721
Operations/Add fault element	ID_WSDL_OPERATIONS_ADDFAULTFUNCTION	33718
Operations/Delete input/output/fault element	ID_WSDL_OPERATIONS_DELETEINPUTOUTPUTFUNCTION	33723
Operations/Add new message to intput/output/fault element	ID_WSDL_OPERATIONS_ADDNEWMESSAGETOTHISELEMENT	33720
Operations/empty operation	ID_WSDL_OPERATIONS_APPENDAOPERATIONTOTHISPORTTYPE	33722
PortType/Insert portType	ID_WSDL_PORTTYPE_INSERTAPORTTYPE	33727
PortType/Delete PortType	ID_WSDL_PORTTYPE_DELETETHISPORTTYPE	33726
Binding/Insert binding	ID_WSDL_BINDING_NEWBINDING	33713
Binding/Delete binding	ID_WSDL_BINDING_DELETEBINDING	33711
Binding/Append Child/soap:body	ID_WSDL_BINDING_APPENDEXTENSIBILITY_SOAPBODY	33706
Binding/Append Child/soap:header	ID_WSDL_BINDING_APPENDEXTENSIBILITY_SOAPHEADER	33708
Binding/Append Child/soap:headerfault	ID_WSDL_BINDING_APPENDEXTENSIBILITY_SOAPHEADERFAULT	33709
Binding/Append Child/soap:fault	ID_WSDL_BINDING_APPENDEXTENSIBILITY_SOAPFAULT	33707
Binding/Append Child/mime:content	ID_WSDL_BINDING_APPENDEXTENSIBILITY_MIMECONTENT	33702

Binding/Append Child/mime:multipartrelated	ID_WSDL_BINDING_APPENDEXTENSIBILITY_MIMEMULTIPARTRELATED	33704
Binding/Append Child/mime:part	ID_WSDL_BINDING_APPENDEXTENSIBILITY_MIMEPART	33705
Binding/Append Child/mime:mimeXml	ID_WSDL_BINDING_APPENDEXTENSIBILITY_MIMEMIMEXML	33703
Binding/Append Child/http:urlencoded	ID_WSDL_BINDING_APPENDEXTENSIBILITY_HTTPURLENCODED	33700
Binding/Append Child/http:urlreplacement	ID_WSDL_BINDING_APPENDEXTENSIBILITY_HTTPURLREPLACEMENT	33701
Binding/Delete extensibility element	ID_WSDL_BINDING_DELETEEXTESIBILITY	33712
Service/Insert service	ID_WSDL_SERVICE_INSERTSERVICE	33731
Service/Delete service	ID_WSDL_SERVICE_DELETETHISSERVICE	33729
Service/Insert port	ID_WSDL_SERVICE_INSERTNEWPORT	33730
Service/Delete port	ID_WSDL_SERVICE_DELETETHISPORT	33728
Types/New schema	ID_WSDL_TYPES_NEWSCHEMA	33733
Types/Edit schema(s) in Schema View	ID_WSDL_TYPES_EDITTHISSCHEMA	33732
Save Diagram...	IDC_WSDL_SAVE_DIAGRAM	34001
Generate Documentation...	ID_WSDL_GENERATEDOCUMENTATION	34233
Reparse WSDL document	IDC_WSDL_REPARSE	33774

6.3.13 SOAPMenu

Commands from the SOAP menu:

Menu Text	Command Name	ID
Create new SOAP request	ID_SOAP_GENERATESOAPMESSAGE	34224
Send request to server	ID_SOAP_SENDREQUESTTOSERVER	34225
Change SOAP request parameters	ID_SOAP_SOAPREQUESTSETTINGS	34227
Soap Debugger Session	ID_SOAP_SOAPDEBUGGER	34226
Go	ID_SOAPDEBUGGER_BUTTONPLAY	34221
Single Step	ID_SOAPDEBUGGER_SINGLESTEP	34222
Break on next Request	ID_SOAPDEBUGGER_BREAKONNEXTREQUEST	34219
Break on next Response	ID_SOAPDEBUGGER_BREAKONNEXTRESPONSE	34220
Stop the proxy server	ID_SOAPDEBUGGER_STOPSERVER	34223
Soap Debugger Options	ID_SOAPDEBUGGEROPTIONS	34218

6.3.14 Tools Menu

Commands from the Tools menu:

© 2007 Altova GmbH

Menu Text	Command Name	ID
Spelling...	IDC_SPELL_CHECK	34154
Spelling options...	IDC_SPELL_OPTIONS	34155
Switch to Scripting environment...	ID_WINDOW_SWITCHTOVBA	34231
Show macros...	ID_WINDOW_VBAMACROS	34232
Project/Assign Scripts to Project	ID_SCRIPTINGPRJ_ASSIGN	34214
Project/Unassign Scripts from Project	ID_SCRIPTINGPRJ_UNASSIGN	34215
Project/Project Scripts active	ID_SCRIPTINGPRJ_ACTIVE	34213
Compare open file with...	ID_XMLDIFF_CHOOSE_FILES	34235
Compare directories...	ID_XMLDIFF_CHOOSE_DIRECTORIES	34234
Compare options...	ID_XMLDIFF_SETTINGS	34236
Customize...	IDC_CUSTOMIZE	34055
Options...	IDC_SETTINGS	33300
<placeholder>	ID_SCRIPTING_MACROITEMS	34249

6.3.15 Window Menu

Commands from the Window menu:

Menu Text	Command Name	ID
Cascade	ID_WINDOW_CASCADE	57650
Tile horizontally	ID_WINDOW_TILE_HORZ	57651
Tile vertically	ID_WINDOW_TILE_VERT	57652
Project window	IDC_PROJECT_WINDOW	34128
Info window	IDC_INFO_WINDOW	34085
Entry Helpers	IDC_ENTRY_HELPERS	34062
Output windows	IDC_OUTPUT_DIALOGBARS	34004
Project and Entry Helpers	IDC_PROJECT_ENTRYHELPERS	34006
All on/off	IDC_ALL_BARS	34031

6.3.16 Help Menu

Commands from the Help menu:

Menu Text	Command Name	ID
Table of Contents...	IDC_HELP_CONTENTS	34076
Index...	IDC_HELP_INDEX	34077
Search...	IDC_HELP_SEARCH	34079
Keyboard Map...	IDC_HELP_KEYMAPDLG	34078
Software Activation...	IDC_ACTIVATION	34005
Order Form...	IDC_OPEN_ORDER_PAGE	34094
Registration...	IDC_REGISTRATION	34131
Check for Updates...	IDC_CHECK_FOR_UPDATES	34275

Support Center...	IDC_OPEN_SUPPORT_PAGE	34096
FAQ on the Web...	IDC_SHOW_FAQ	34153
Download Components and Free Tools...	IDC_OPEN_COMPONENTS_PAGE	34093
XMLSpy on the Internet..	IDC_OPEN_XML_SPY_HOME	34098
XMLSpy Training...	ID_HELP_XMLSPYTRAINING	34210
About XMLSpy...	ID_APP_ABOUT	57664

© 2007 Altova GmbH

6.4 Accessing XMLSpy API

The focus of this documentation is the ActiveX controls and interfaces required to integrate the XMLSpy user interface into your application. To allow you to automate or control the functionality of the integrated components, the following properties give you access to the XMLSpy automation interface (XMLSpy API):

`XMLSpyControl.Application`
`XMLSpyControlDocument.Document`
`XMLSpyControlPlaceHolder.Project`

Some restrictions apply to the usage of the XMLSpy automation interface when integrating XMLSpyControl at document-level. See Integration at document level for details.

6.5 Object Reference

Objects:
Command
Commands
XMLSpyControl
XMLSpyControlDocument
XMLSpyControlPlaceHolder

To give access to standard XMLSpy functionality, objects of the **XMLSpy automation interface** can be accessed as well. See XMLSpyControl.Application, XMLSpyControlDocument.Document and XMLSpyControlPlaceHolder.Project for more information.

6.5.1 Command

Properties:
ID
Label
IsSeparator
ToolTip
StatusText
Accelerator
SubCommands

Description:
Each Command object can be one of three possible types:

- **Command**: ID is set to a value greater 0 and Label is set to the command name. IsSeparator is false and the SubCommands collection is empty.
- **Separator**: IsSeparator is true. ID is 0 and Label is not set. The SubCommands collection is empty.
- (**Sub**) **Menu**: The SubCommands collection contains Command objects and Label is the name of the menu. ID is set to 0 and IsSeparator is false.

Accelerator

Property: Label as string

Description:
For command objects that are children of the ALL_COMMANDS collection, this is the accelerator key defined for the command. If the command has no accelerator key assigned, this property returns the empty string.

The string representation of the accelerator key has the following format:

```
[ALT+][CTRL+][SHIFT+]key
```

Where key is converted using the Windows Platform SDK function GetKeyNameText.

ID

Property: ID as long

© 2007 Altova GmbH

Description:
ID is 0 for separators and menus.
For commands, this is the ID which can be used with Exec and QueryStatus.

IsSeparator

Property: IsSeparator as boolean

Description:
True if the command is a separator.

Label

Property: Label as string

Description:
Label is empty for separators.
For command objects that are children of the ALL_COMMANDS collection, this is a unique name. Command icons are stored in files with this name. See Query Commands for more information.

For command objects that are children of menus, the label property holds the command's menu text.
For sub-menus, this property holds the menu text.

StatusText

Property: Label as string

Description:
For command objects that are children of the ALL_COMMANDS collection, this is the text shown in the status bar when the command is selected.

SubCommands

Property: SubCommands as Commands

Description:
The SubCommands collection holds any sub-commands if this command is actually a menu or submenu.

ToolTip

Property: ToolTip as string

Description:
For command objects that are children of the ALL_COMMANDS collection, this is the text shown as tool-tip.

6.5.2 Commands

Properties:
Count

Item

Description:
Collection of Command objects to get access to command labels and IDs of the XMLSpyControl. Those commands can be executed with the Exec method and their status can be queried with QueryStatus.

Count

Property: Count as long

Description:
Number of Command objects on this level of the collection.

Item

Property: Item (n as long) as Command

Description:
Gets the command with the index n in this collection. Index is 1-based.

6.5.3 XMLSpyControl

Properties:
IntegrationLevel
Appearance
Application
BorderStyle
CommandsList
CommandsStructure (deprecated)
EnableUserPrompts
MainMenu
Toolbars

Methods:
Open
Exec
QueryStatus

Events:
OnUpdateCmdUI
OnOpenedOrFocused
OnCloseEditingWindow
OnFileChangedAlert
OnContextChanged
OnDocumentOpened
OnValidationWindowUpdated

This object is a complete ActiveX control and should only be visible if the XMLSpy library is used in the Application Level mode.

CLSID: a258bba2-3835-4c16-8590-72b44f52c471
ProgID: Altova.XMLSpyControl

© 2007 Altova GmbH

Properties

The following properties are defined:

IntegrationLevel
EnableUserPrompts
Appearance
BorderStyle

Command related properties:
CommandsList
MainMenu
Toolbars
CommandsStructure (deprecated)

Access to XMLSpyAPI:
Application

Appearance

Property: `Appearance` as short

Dispatch Id: *-520*

Description:
A value not equal to `0` displays a client edge around the control. Default value is `0`.

Application

Property: `Application` as Application

Dispatch Id: *1*

Description:
The `Application` property gives access to the `Application` object of the complete XMLSpy automation server API. The property is read-only.

BorderStyle

Property: `BorderStyle` as short

Dispatch Id: *-504*

Description:
A value of `1` displays the control with a thin border. Default value is `0`.

CommandsList

Property: `CommandList` as Commands (read-only)

Dispatch Id: *1004*

Description:

This property returns a flat list of all commands defined available with XMLSpyControl. For more information see C# Sample.

CommandsStructure (deprecated)

Property: `CommandsStructure` as `Commands` (deprecated)

Dispatch Id: *3*

Remark:
This property is deprecated. Instead, use `CommandsList`, `MainMenu`, `Toolbars`.

Description:
The `CommandsStructure` collection contains all commands of the XMLSpyControl as `Command` objects. At the first level of the collection two special `Command` objects with the following labels are accessible:

- `IDR_XMLSPY`: This object holds all commands as hierarchical menu structure.
- `ALL_COMMANDS`: This object holds all commands in a flat list.

Sample:
C# code to access the first level of the collection.

```
Commands    objCommands;
objCommands = axXMLSpyControl.CommandsStructure;

long nCount = objCommands.Count;

for(long idx = 0;idx < nCount;idx++)
{
    Command objCommand;
    objCommand = objCommands[(int)idx];

    // We are looking for the Menu with the name IDR_XMLSPY. This
    menu should contain
    // the complete main menu of XMLSpy.

    if(objCommand.Label == "IDR_XMLSPY")
    {
        // read menu structure here...
    }

    if(objCommand.Label == "ALL_COMMANDS")
    {
        // read all commands here...
    }
}
```

EnableUserPrompts

Property: `EnableUserPrompts` as boolean

Dispatch Id: *1006*

© 2007 Altova GmbH

Description:
Setting this property to *false*, disables user prompts in the control. The default value is *true*.

IntegrationLevel

Property: `IntegrationLevel` as `ICActiveXIntegrationLevel`

Dispatch Id: *1000*

Description:
The `IntegrationLevel` property determines the operation mode of the control. See also Integration at the application level and Integration at document level for more information.

Note: It is important to set this property immediately after the creation of the `XMLSpyControl` object.

MainMenu

Property: `MainMenu` as `Command`(read-only)

Dispatch Id: *1003*

Description:
This property gives access to the description of the XMLSpyControl main menu. For more information see C# Sample.

Toolbars

Property: `Toolbars` as `Commands`(read-only)

Dispatch Id: *1005*

Description:
This property returns a list of all toolbar descriptions that describe all toolbars available with XMLSpyControl. For more information see C# Sample.

Methods

The following methods are defined:

`Open`
`Exec`
`QueryStatus`

Exec

Method: `Exec` (`nCmdID` as long) as boolean

Dispatch Id: *6*

Description:

Exec calls the XMLSpy command with the ID nCmdID. If the command can be executed, the method returns true. See also CommandsStructure to get a list of all available commands and QueryStatus to retrieve the status of any command.

Open

Method: Open (strFilePath as string) as boolean

Dispatch Id: *5*

Description:
The result of the method depends on the extension passed in the argument strFilePath. If the file extension is .mfd, a new document is opened. If the file extension is .mfp, the corresponding project is opened. If a different file extension is passed into the method, the control tries to load the file as a new component into the active document.

Do not use this method to load documents or projects when using the control in document-level integration mode. Instead, use XMLSpyControlDocument.Open and XMLSpyControlPlaceHolder.OpenProject.

QueryStatus

Method: QueryStatus (nCmdID as long) as long

Dispatch Id: *7*

Description:
QueryStatus returns the enabled/disabled and checked/unchecked status of the command specified by nCmdID. The status is returned as a bit mask.

Bit	Value	Name	Meaning
0	1	Supported	Set if the command is supported.
1	2	Enabled	Set if the command is enabled (can be executed).
2	4	Checked	Set if the command is checked.

This means that if QueryStatus returns 0 the command ID is not recognized as a valid XMLSpy command. If QueryStatus returns a value of 1 or 5, the command is disabled.

Events

The XMLSpyControl ActiveX control provides the following connection point events:

OnUpdateCmdUI
OnOpenedOrFocused
OnCloseEditingWindow
OnFileChangedAlert
OnContextChanged

OnDocumentOpened
OnValidationWindowUpdated

© 2007 Altova GmbH

OnCloseEditingWindow

Event: `OnCloseEditingWindow` (i_strFilePath as String) as boolean

Dispatch Id: *1002*

Description:
This event is triggered when XMLSpy needs to close an already open document. As an answer to this event, clients should close the editor window associated with *i_strFilePath*. Returning *true* from this event indicates that the client has closed the document. Clients can return *false* if no specific handling is required and XMLSpyControl should try to close the editor and destroy the associated document control.

OnContextChanged

Event: `OnContextChanged` (`i_strContextName` as String, `i_bActive` as bool) as bool

Dispatch Id: *1004*

Description:
This event is triggered when XMLSpy activates or de-actives one of the following operational contexts:

- XSLT Profiling - `"XSLTProfiling"` is passed as the context name
- XSLT / XQuery debugging - `"DebuggingXSLT"` is passed as the context name
- SOAP debugging - `"DebuggingSOAP"` is passed as the context name

OnDocumentOpened

Event: `OnDocumentOpened` (`objDocument` as Document)

Dispatch Id: *1*

Description:
This event is triggered whenever a document is opened. The argument `objDocument` is a `Document` object from the XMLSpy automation interface and can be used to query for more details about the document, or perform additional operations. When integrating on document-level, it is often better to use the event `XMLSpyControlDocument.OnDocumentOpened` instead.

OnFileChangedAlert

Event: `OnFileChangedAlert` (`i_strFilePath` as String) as bool

Dispatch Id: *1001*

Description:
This event is triggered when a file loaded with XMLSpyControl, is changed on the harddisk by another application. Clients should return true, if they handled the event, or false, if XMLSpy should handle it in its customary way, i.e. prompting the user for reload.

OnLicenseProblem

Event: `OnLicenseProblem` (`i_strLicenseProblemText` as String)

Dispatch Id: *1005*

Description:
This event is triggered when XMLSpyControl detects that no valid license is available for this control. In case of restricted user licenses this can happen some time after the control has been initialized. Integrators should use this event to disable access to this control's functionality. After returning from this event, the control will block access to its functionality (e.g. show empty windows in its controls and return errors on requests).

OnOpenedOrFocused

Event: `OnOpenedOrFocused` (`i_strFilePath` as String, `i_bOpenWithThisControl` as bool)

Dispatch Id: *1000*

Description:
When integrating at application level, this event informs clients that a document has been opened, or made active by XMLSpy.

When integrating at document level, this event instructs the client to open the file `i_strFilePath` in a document window. If the file is already open, the corresponding document window should be made the active window.

if `i_bOpenWithThisControl` is `true`, the document must be opened with XMLSpyControl, since internal access is required. Otherwise, the file can be opened with different editors.

OnUpdateCmdUI

Event: `OnUpdateCmdUI ()`

Dispatch Id: *1003*

Description:
Called frequently to give integrators a good opportunity to check status of XMLSpy commands using `XMLSpyControl.QueryStatus`. Do not perform long operations in this callback.

OnValidationWindowUpdated

Event: `OnValidationWindowUpdated` ()

Dispatch Id: *3*

Description:
This event is triggered whenever the validation output window, is updated with new information.

6.5.4 XMLSpyControlDocument

Properties:
`Appearance`
`BorderStyle`
`Document`
`IsModified`
`Path`
`ReadOnly`

© 2007 Altova GmbH

Methods:
Exec
New
Open
QueryStatus
Reload
Save
SaveAs

Events:
OnDocumentOpened
OnDocumentClosed
OnModifiedFlagChanged
OnContextChanged
OnFileChangedAlert
OnActivate

If the XMLSpyControl is integrated in the Document Level mode each document is displayed in an own object of type `XMLSpyControlDocument`. The `XMLSpyControlDocument` contains only one document at the time but can be reused to display different files one after another.

This object is a complete ActiveX control.

```
CLSID: 52A552E6-2AB8-4e3e-B545-BE998233DDA0
ProgID: Altova.XMLSpyControlDocument
```

Properties

The following properties are defined:

ReadOnly
IsModified
Path
Appearance
BorderStyle

Access to XMLSpyAPI:
Document

Appearance

Property: `Appearance` as short

Dispatch Id: *-520*

Description:
A value not equal to `0` displays a client edge around the document control. Default value is `0`.

BorderStyle

Property: `BorderStyle` as short

Dispatch Id: *-504*

Description:
A value of `1` displays the control with a thin border. Default value is `0`.

Document

Property: `Document` as Document

Dispatch Id: *1*

Description:
The `Document` property gives access to the `Document` object of the XMLSpy automation server API. This interface provides additional functionalities which can be used with the document loaded in the control. The property is read-only.

IsModified

Property: `IsModified` as boolean (read-only)

Dispatch Id: *1006*

Description:
`IsModified` is *true* if the document content has changed since the last open, reload or save operation. It is *false*, otherwise.

Path

Property: `Path` as string

Dispatch Id: *1005*

Description:
Sets or gets the full path name of the document loaded into the control.

ReadOnly

Property: `ReadOnly` as boolean

Dispatch Id: *1007*

Description:
Using this property you can turn on and off the read-only mode of the document. If `ReadOnly` is `true` it is not possible to do any modifications.

Methods

The following methods are defined:

Document handling:
New
Open
Reload
Save
SaveAs

© 2007 Altova GmbH

Command Handling:
Exec
QueryStatus

Exec

Method: Exec (nCmdID as long) as boolean

Dispatch Id: *8*

Description:
Exec calls the XMLSpy command with the ID nCmdID. If the command can be executed, the method returns true. The client should call the Exec method of the document control if there is currently an active document available in the application.

See also CommandsStructure to get a list of all available commands and QueryStatus to retrieve the status of any command.

New

Method: New () as boolean

Dispatch Id: *1000*

Description:
This method initializes a new document inside the control..

Open

Method: Open (strFileName as string) as boolean

Dispatch Id: *1001*

Description:
Open loads the file strFileName as the new document into the control.

QueryStatus

Method: QueryStatus (nCmdID as long) as long

Dispatch Id: *9*

Description:
QueryStatus returns the enabled/disabled and checked/unchecked status of the command specified by nCmdID. The status is returned as a bit mask.

Bit	Value	Name	Meaning
0	1	Supported	Set if the command is supported.

1	2	Enabled	Set if the command is enabled (can be executed).
2	4	Checked	Set if the command is checked.

This means that if `QueryStatus` returns `0` the command ID is not recognized as a valid XMLSpy command. If `QueryStatus` returns a value of `1` or `5` the command is disabled. The client should call the `QueryStatus` method of the document control if there is currently an active document available in the application.

Reload

Method: `Reload` () as boolean

Dispatch Id: *1002*

Description:
`Reload` updates the document content from the file system.

Save

Method: `Save` () as boolean

Dispatch Id: *1003*

Description:
`Save` saves the current document at the location `Path`.

SaveAs

Method: `OpenDocument` (`strFileName` as string) as boolean

Dispatch Id: *1004*

Description:
`SaveAs` sets `Path` to *strFileName* and then saves the document to this location.

Events

The XMLSpyControlDocument ActiveX control provides following connection point events:

`OnDocumentOpened`
`OnDocumentClosed`
`OnModifiedFlagChanged`
`OnContextChanged`
`OnFileChangedAlert`
`OnActivate`
OnSetEditorTitle

OnActivate

Event: `OnActivate` ()

Dispatch Id: *1005*

© 2007 Altova GmbH

Description:
This event is triggered when the document control is activated, has the focus, and is ready for user input.

OnContextChanged

Event: `OnContextChanged` (`i_strContextName` as String, `i_bActive` as bool) as bool

Dispatch Id: *1004*

Description:
This event is triggered when this document is shown in a different XMLSpy view. The following values are passed:

- Grid view - `"View_0"` is passed as the context name
- Text view - `"View_1"` is passed as the context name
- Browser view - `"View_2"` is passed as the context name
- Schema view - `"View_3"` is passed as the context name
- Authentic view - `"View_4"` is passed as the context name
- WSDL view - `"View_5"` is passed as the context name

OnDocumentClosed

Event: `OnDocumentClosed` (`objDocument` as Document)

Dispatch Id: *1001*

Description:
This event is triggered whenever the document loaded into this control is closed. The argument `objDocument` is a `Document` object from the XMLSpy automation interface and should be used with care.

OnDocumentOpened

Event: `OnDocumentOpened` (`objDocument` as Document)

Dispatch Id: *1000*

Description:
This event is triggered whenever a document is opened in this control. The argument `objDocument` is a `Document` object from the XMLSpy automation interface, and can be used to query for more details about the document, or perform additional operations.

OnDocumentSaveAs

Event: `OnContextDocumentSaveAs` (`i_strFileName` as String)

Dispatch Id: *1007*

Description:
This event is triggered when this document gets internally saved under a new name.

OnFileChangedAlert

Event: `OnFileChangedAlert` () as bool

Dispatch Id: *1003*

Description:
This event is triggered when the file loaded into this document control, is changed on the harddisk by another application. Clients should return true, if they handled the event, or false, if XMLSpy should handle it in its customary way, i.e. prompting the user for reload.

OnModifiedFlagChanged

Event: `OnModifiedFlagChanged` (`i_bIsModified` as boolean)

Dispatch Id: *1002*

Description:
This event gets triggered whenever the document changes between modified and unmodified state. The parameter *i_bIsModifed* is *true* if the document contents differs from the original content, and *false*, otherwise.

OnSetEditorTitle

Event: OnSetEditorTitle ()

Dispatch Id: *1006*

Description:
This event is being raised when the contained document is being internally renamed.

6.5.5 XMLSpyControlPlaceHolder

Properties available for all kinds of placeholder windows:
`PlaceholderWindowID`

Properties for project placeholder window:
`Project`

Methods for project placeholder window:
`OpenProject`
`CloseProject`

The `XMLSpyControlPlaceHolder` control is used to show the additional XMLSpy windows like Overview, Library or Project window. It is used like any other ActiveX control and can be placed anywhere in the client application.

```
CLSID: 135DEEF4-6DF0-47c2-8F8C-F145F5F3F672
ProgID: Altova.XMLSpyControlPlaceHolder
```


© 2007 Altova GmbH

Properties

The following properties are defined:

PlaceholderWindowID

Access to XMLSpyAPI:
Project

Label

Property: Label as String (read-only)

Dispatch Id: *1001*

Description:
This property gives access to the title of the placeholder. The property is read-only.

PlaceholderWindowID

Property: PlaceholderWindowID as XMLSpyControlPlaceholderWindow

Dispatch Id: *1*

Description:
Using this property the object knows which XMLSpy window should be displayed in the client area of the control. The PlaceholderWindowID can be set at any time to any valid value of the XMLSpyControlPlaceholderWindow enumeration. The control changes its state immediately and shows the new XMLSpy window.

Project

Property: Project as Project (read-only)

Dispatch Id: *2*

Description:
The Project property gives access to the Project object of the XMLSpy automation server API. This interface provides additional functionalities which can be used with the project loaded into the control. The property will return a valid project interface only if the placeholder window has PlaceholderWindowID with a value of XMLSpyXProjectWindow (=3). The property is read-only.

Methods

The following method is defined:

OpenProject
CloseProject

OpenProject

Method: `OpenProject` (`strFileName` as string) as boolean

Dispatch Id: *3*

Description:
`OpenProject` loads the file `strFileName` as the new project into the control. The method will fail if the placeholder window has a `PlaceholderWindowID` different to `XMLSpyXProjectWindow (=3)`.

CloseProject

Method: `CloseProject` ()

Dispatch Id: *4*

Description:
`CloseProject` closes the project loaded the control. The method will fail if the placeholder window has a `PlaceholderWindowID` different to `XMLSpyXProjectWindow (=3)`.

Events

The XMLSpyControlPlaceholder ActiveX control provides following connection point events:

`OnModifiedFlagChanged`

OnModifiedFlagChanged

Event: `OnModifiedFlagChanged` (`i_bIsModified` as boolean)

Dispatch Id: *1*

Description:
This event gets triggered only for placeholder controls with a `PlaceholderWindowID` of `XMLSpyXProjectWindow (=3)`. The event is fired whenever the project content changes between modified and unmodified state. The parameter *i_bIsModifed* is *true* if the project contents differs from the original content, and *false*, otherwise.

6.5.6 Enumerations

The following enumerations are defined:

`ICActiveXIntegrationLevel`
`XMLSpyControlPlaceholderWindow`

ICActiveXIntegrationLevel

Possible values for the IntegrationLevel property of the XMLSpyControl.

```
ICActiveXIntegrationOnApplicationLevel    = 0
```

 © 2007 Altova GmbH

```
ICActiveXIntegrationOnDocumentLevel       = 1
```

XMLSpyControlPlaceholderWindow

This enumeration contains the list of the supported additional XMLSpy windows.

```
XMLSpyControlNoToolWnd                        = -1,
XMLSpyControlEntryHelperTopToolWnd            = 0,
XMLSpyControlEntryHelperMiddleToolWnd         = 1,
XMLSpyControlEntryHelperBottomToolWnd         = 2,
XMLSpyControlValidatorOutputToolWnd           = 3,
XMLSpyControlProjectWindowToolWnd             = 4,
XMLSpyControlXSLTDebuggerContextToolWnd       = 5,
XMLSpyControlXSLTDebuggerCallstackToolWnd     = 6,
XMLSpyControlXSLTDebuggerVariableToolWnd      = 7,
XMLSpyControlXSLTDebuggerWatchToolWnd         = 8,
XMLSpyControlXSLTDebuggerTemplateToolWnd      = 9,
XMLSpyControlXSLTDebuggerInfoToolWnd          = 10,
XMLSpyControlXSLTDebuggerMessageToolWnd       = 11,
XMLSpyControlXSLTDebuggerTraceToolWnd         = 12,
XMLSpyControlSOAPDebuggerToolWnd              = 13,
XMLSpyControlXPathProfilerListToolWnd         = 14,
XMLSpyControlXPathProfilerTreeToolWnd         = 15,
XMLSpyControlXPathDialogToolWnd               = 16
```

Altova XMLSpy 2007 Enterprise Edition

Anhänge

Anhänge

Diese Anhänge enthalten technische Informationen über XMLSpy und wichtige Lizenzinformationen. Die einzelnen Anhänge sind in die folgenden Unterabschnitte gegliedert:

Technische Daten

- Betriebssystem und erforderlicher Arbeitsspeicher
- Altova XML Parser
- Altova XSLT- und XQuery-Prozessor
- Unicode-Unterstützung
- Internet-Verwendung
- Lizenzüberwachung

Lizenzinfomationen

- Electronic Software Distribution (EDI)
- Copyright
- Endbenutzer-Lizenzvereinbarung (EULA)

© 2007 Altova GmbH

1 Prozessor-Informationen

Dieser Abschnitt enthält Informationen über implementierungsspezifische Funktionen des Altova XSLT 1.0-Prozessors, Altova XSLT 2.0-Prozessors und des Altova XQuery 1.0-Prozessors.

1.1 XSLT 1.0-Prozessor: Implementierungsinformationen

Der Altova XSLT 1.0-Prozessor ist in die Altova XML-Produkte XMLSpy, StyleVision, Authentic und MapForce integriert. Außerdem steht er im kostenlosen AltovaXML-Paket zur Verfügung. Im Altova XSLT 1.0-Prozessor sind die W3C-Spezifikationen implementiert und entsprechen der XSLT 1.0 Recommendation vom 16. November 1999 und der XPath 1.0 Recommendation vom 16. November 1999. Im Folgenden sind Einschränkungen und implementierungsspezifisches Verhalten aufgelistet.

Einschränkungen

- Die Elemente **xsl:preserve-space** und **xsl:strip-space** werden nicht unterstützt.
- Wenn das Attribut `method` von `xsl:output` auf HTML gesetzt ist, oder wenn standardmäßig HTML output ausgewählt ist, werden Sonderzeichen in der XML- oder XSLT-Datei direkt als Sonderzeichen in die HTML-Dokument eingefügt und nicht als HTML-Zeichenreferenzen in der Ausgabe. So wird z.B. das Zeichen ` ` (die Dezimalzeichenreferenz für ein geschütztes Leerzeichen) nicht als ` ` in den HTML-Code eingefügt sondern direkt als geschütztes Leerzeichen.

Behandlung der Implementierung von Nur-Whitespace Nodes in XML-Quelldokumenten
Aus den XML-Daten (und somit dem XML-Infoset), die an den Altova XSLT 1.0-Prozessor übergeben werden, werden boundary-whitespace-only Text Nodes entfernt. (Ein boundary-whitespace-only Text Node ist ein Child-Textnode, der nur Whitespaces enthält und der zwischen zwei Elementen innerhalb eines mixed-content-Elements vorkommt). Dies kann sich auf den Wert auswirken, der von den Funktionen `fn:position()`, `fn:last()` und `fn:count()` zurückgegeben wird.

In jeder Node-Auswahl, bei der auch Text Nodes ausgewählt werden, sind normalerweise auch Boundary-whitespace-only Text Nodes enthalten. Da jedoch beim XML Infoset, das von den Altova-Prozessoren verwendet wird, boundary-whitespace-only Text Nodes entfernt werden, sind diese Nodes im XML Infoset nicht vorhanden. Folglich unterscheidet sich die Größe der Auswahl und die Nummierung der Nodes von einer, in der diese Text Nodes enthalten sind. Aus diesem Grund können sich die Ergebnisse der Funktionen `fn:position()`, `fn:last()` und `fn:count()` von denen anderer Prozessoren unterscheiden.

Am häufigsten tritt die Situation, dass boundary-whitespace-only Text Nodes als gleichrangige Elemente anderer Elemente überprüft werden, dann auf, wenn `xsl:apply-templates` verwendet wird, um Templates anzuwenden. Wenn die Funktionen `fn:position()`, `fn:last()` und `fn:count()` in Patterns mit einer Namensüberprüfung verwendet werden (z.B. `para[3], kurz für para[position()=3]`), sind boundary-whitespace-only Nodes irrelevant, da nur die benannten Elemente (`para` im obigen Beispiel) ausgewählt werden. (Beachten Sie jedoch, dass boundary-whitespace-only Nodes in Patterns, in denen ein Platzhalterzeichen wie z.B. `*[10]` verwendet wird, sehr wohl relevant sind).

Hinweis: Wenn der boundary-whitespace-only Text Node in der Ausgabe benötigt wird, fügen Sie das erforderliche Whitespace-Zeichen in eines der benachbarten Child-Elemente ein. So erzeugt z.B. das XML-Fragment:

```
<para>This is <b>bold</b> <i>italic</>.</para>
```

bei Verarbeitung mit dem XSLT Template

```
<xsl:template match="para">
   <xsl:apply-templates/>
</xsl:template>
```

 © 2007 Altova GmbH

folgendes Resultat:

```
This is bolditalic.
```

Um in der Ausgabe ein Leerzeichen zwischen `bold` und `italic` zu erhalten, fügen Sie entweder in das Element `<b>` oder `<i>` in der XML-Quelle ein Leerzeichen ein. z.B.:

```
<para>This is <b>bold</b> <i> italic</>.</para> oder
<para>This is <b>bold&#x20;</b> <i>italic</>.</para> oder
<para>This is <b>bold</b><i>&#x20;italic</>.</para>
```

Wenn nun ein solches XML-Fragment mit demselben XSLT Template verarbeitet wird, erhalten Sie folgendes Ergebnis:

```
This is bold italic.
```

1.2 XSLT 2.0-Prozessor: Implementierungsinformationen

Der Altova XSLT 2.0-Prozessor ist in die Altova XML-Produkte XMLSpy, StyleVision, Authentic und MapForce integriert. Außerdem steht er im kostenlosen AltovaXML-Paket zur Verfügung. In diesem Abschnitt werden die implementierungsspezifischen Aspekte des Prozessors beschrieben. Im ersten Teil des Abschnitts finden Sie allgemeine Informationen über den Prozessor. Im Anschluss daran finden Sie eine Liste der implementierungsspezifischen Aspekte der XSLT 2.0-Funktionen.

Informationen über das implementierungsspezifische Verhalten von XPath 2.0-Funktionen finden Sie im Abschnitt XPath 2.0- und XQuery 1.0-Funktionen.

1.2.1 Allgemeine Informationen

Der Altova XSLT 2.0-Prozessor entspricht der XSLT 2.0 Recommendation vom 23. Jänner 2007. Beachten Sie bitte die folgenden allgemeinen Informationen über den Prozessor.

Rückwärtskompatibilität
Der Altova XSLT 2.0-Prozessor ist rückwärtskompatibel. Die Rückwärtskompatibilität des XSLT 2.0.-Prozessors kommt nur dann zum Einsatz, wenn Sie den XSLT 2.0-Prozessor von Altova XML zur Verarbeitung eines XSLT 1.0 Stylesheets verwenden. Beachten Sie, dass sich das Ergebnis des XSLT 1.0-Prozessors und des rückwärtskompatiblen XSLT 2.0.Prozessors unter Umständen unterscheiden kann.

Bei allen anderen Altova-Produkten wird Rückwärtskompatibilität nie benötigt, da automatisch der entsprechende Prozessor für die Transformation ausgewählt wird. Angenommen, Sie legen fest, dass ein bestimmtes XML-Dokument in XMLSpy mit einem XSLT 1.0 Stylesheet verarbeitet werden soll. Bei Aufruf des Transformationsbefehls wählt XMLSpy zur Transformation automatisch den XSLT 1.0-Prozessor aus.

Anmerkung:
Diese Auswahl basiert auf dem Wert im Attribut `version` des Stylesheet-Elements `stylesheet` oder `transform`.

Namespaces
In Ihrem XSLT 2.0 Stylesheet sollten die folgenden Namespaces deklariert sein, damit Sie die in XSLT 2.0 verfügbaren Typ-Konstruktoren und Funktionen verwenden können. Normalerweise werden die unten aufgelisteten Präfixe verwendet; bei Bedarf können Sie auch andere Präfixe verwenden.

Namespace Name	Präfix	Namespace URI
XML Schema-Typen	`xs:`	`http://www.w3.org/2001/XMLSchema`
XPath 2.0-Funktionen	`fn:`	`http://www.w3.org/2005/xpath-functions`

Normalerweise werden diese Namespaces im Element `xsl:stylesheet` oder `xsl:transform` deklariert, wie unten gezeigt:

```
<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"
```


© 2007 Altova GmbH

```
   ...
</xsl:stylesheet>
```

Beachten Sie bitte die folgenden Punkte:

- Der Altova XSLT 2.0-Prozessor verwendet als Default Functions Namespace den Namespace für XPath 2.0- und XQuery 1.0-Funktionen (siehe Tabelle oben). Sie können daher XPath 2.0- und XSLT 2.0-Funktionen in Ihrem Stylesheet ohne Präfix verwenden. Wenn Sie den Namespace für XPath 2.0-Funktionen in Ihrem Stylesheet mit einem Präfix deklarieren, können Sie zusätzlich dazu das in der Deklaration zugewiesene Präfix verwenden.
- Bei Verwendung von Typ-Konstrukturoen und Typen aus dem XML Schema-Namespace, muss bei Aufruf des Typ-Konstruktors (z.B. xs:date) das in der jeweiligen Namespace-Deklaration verwendeten Präfix verwendet werden.
- In den Candidate Recommendations vom 23. Jänner 2007 wurden die Datentypen untypedAtomic und duration (dayTimeDuration und yearMonthDuration), die sich zuvor im XPath Datatypes Namespace befanden (normalerweise mit dem Präfix xdt:) in den XML-Schema-Namespace verschoben.
- Einige XPath 2.0-Funktionen haben denselben Namen wie XML Schema-Datentypen. So gibt es z.B. für die XPath-Funktionen fn:string und fn:boolean XML-Schema-Datentypen mit demselben lokalen Namen: xs:string und xs:boolean. Wenn Sie daher den XPath-Ausdruck string('Hello') verwenden, wird der Ausdruck als fn:string('Hello') ausgewertet und nicht als xs:string('Hello').

Schemafähigkeit
Der Altova XSLT 2.0-Prozessor ist schemafähig.

Whitespaces im XML-Dokument
Standardmäßig entfernt der Altova XSLT 2.0-Prozessor alle boundary whitespaces aus boundary-whitespace-only Nodes im XML-Quelldokument. Dies wirkt sich auf die Werte aus, die die Funktionen `fn:position()`, `fn:last()`, `fn:count()` und `fn:deep-equal()` zurückgeben. Nähere Informationen dazu finden Sie in den Abschnitten über XPath 2.0- und XQuery 1.0-Funktionen unter Nur-Whitespace Nodes im XML-Quelldokument.

Hinweis: Wenn in der Ausgabe ein boundary-whitespace-only-Text Node erforderlich ist, fügen Sie das gewünschte Whitespace-Zeichen in eines der beiden benachbarten Child-Elemente ein. So erzeugt z.B. das XML-Fragment:

```
<para>This is <b>bold</b> <i>italic</>.</para>
```

bei Verarbeitung mit dem XSLT Template

```
<xsl:template match="para">
   <xsl:apply-templates/>
</xsl:template>
```

folgendes Resultat:

```
This is bolditalic.
```

Um in der Ausgabe ein Leerzeichen zwischen `bold` und `italic` zu erhalten, fügen Sie entweder in das Element `<b>` oder `<i>` in der XML-Quelle ein Leerzeichen ein. z.B.:

```
<para>This is <b>bold</b> <i> italic</>.</para>  oder
<para>This is <b>bold&#x20;</b> <i>italic</>.</para>  oder
<para>This is <b>bold</b><i>&#x20;italic</>.</para>
```

Wenn nun ein solches XML-Fragment mit demselben XSLT Template verarbeitet wird, erhalten Sie folgendes Ergebnis:

```
This is bold italic.
```

XSLT 2.0-Elemente und -Funktionen
Einschränkungen und implementierungsspezifisches Verhalten von XSLT 2.0-Elementen und -Funktionen werden im Abschnitt XSLT 2.0-Elemente und -Funktionen aufgelistet.

XPath 2.0-Funktionen
Implementierungsspezifisches Verhalten von XPath 2.0-Funktionen wird im Abschnitt XPath 2.0- und XQuery 1.0-Funktionen aufgelistet.

1.2.2 XSLT 2.0-Elemente und -Funktionen

Einschränkungen
Die Elemente **`xsl:preserve-space`** und **`xsl:strip-space`** werden nicht unterstützt.

Implementierungsspezifisches Verhalten
Im Folgenden finden Sie eine Beschreibung, wie der Altova XSLT 2.0-Prozessor implementierungsspezifische Aspekte des Verhaltens bestimmter XSLT 2.0-Funktionen behandelt.

`function-available`
Die Funktion überprüft, ob die XSLT 2.0-Funktionen zur Verfügung stehen und nicht ob die XPath 2.0-Funktionen verfügbar sind.

`unparsed-text`
Das Attribut `href` akzeptiert (i) relative Pfade für Dateien im Basis-URI-Ordner und (ii) absolute Pfade mit oder ohne das `file://`-Protokoll.

© 2007 Altova GmbH

1.3 XQuery 1.0-Prozessor: Implementierungsinformationen

Der Altova XQuery 1.0-Prozessor ist in die Altova XML-Produkte XMLSpy und MapForce integriert. Er steht auch in Form des kostenlosen Pakets AltovaXML zur Verfügung. Dieser Abschnitt enthält Informationen über implementierungsspezifische Aspekte seines Verhaltens.

Standardkonformität
Der Altova XQuery 1.0-Prozessor entspricht den XQuery 1.0 CR vom 23. Jänner 2007. Der Query-Standard stellt frei, wie viele Funktionen zu implementieren sind. Im Folgenden finden Sie eine Liste, wie der Altova XQuery 1.0-Prozessor diese Funktionen implementiert.

Schemafähigkeit
Der Altova XQuery 1.0-Prozessor ist **schemafähig.**

Codierung
Die UTF-8 und die UTF-16 Zeichen-Kodierungen werden unterstützt.

Namespaces
Die folgenden Namespace-URIs und die damit verknüpften Bindings sind vordefiniert.

Namespace-Name	Präfix	Namespace URI
XML Schema-Typen	`xs:`	`http://www.w3.org/2001/XMLSchema`
Schema-Instanz	`xsi:`	`http://www.w3.org/2001/XMLSchema-instance`
Built-in Funktionen	`fn:`	`http://www.w3.org/2005/xpath-functions`
Lokale Funktionen	`local:`	`http://www.w3.org/2005/xquery-local-functions`

Beachten Sie bitte die folgenden Punkte:

- Der Altova XQuery 1.0-Prozessor ist so konfiguriert, dass die oben aufgelisteten Präfixe an die entsprechenden Namespaces gebunden sind.
- Da der oben angeführte Namespace für Built-in Funktionen der Default Functions Namespace in XQuery ist, muss beim Aufruf von built-in-Funktionen das Präfix `fn:` nicht verwendet werden (`string("Hello")` ruft z.B. die Funktion `fn:string` auf). Das Präfix `fn:` kann jedoch verwendet werden, um eine built-in-Funktion aufzurufen, ohne die Namespace im Abfrage-Prolog deklarieren zu müssen (z.B.: `fn:string("Hello")`).
- Sie können den Default Functions Namespace durch Deklarierung des `default function namespace`-Ausdrucks im Abfrageprolog ändern.
- Bei Verwendung von Typen aus dem XML Schema- und dem XPath-Datentypen-Namespace kann das Präfix `xs:` verwendet werden, ohne dass Sie den Namespace explizit deklarieren müssen und dieses Präfix im Abfrageprolog daran binden müssen. (Beispiele: `xs:date` und `xs:yearMonthDuration`.) Wenn Sie ein anderes Präfix verwenden möchten, muss dieses im Abfrageprolog für die Namespaces explizit deklariert werden. (Beispiel `declare namespace alt = "http://www.w3.org/2001/XMLSchema"; alt:date("2004-10-04")`.)
- Beachten Sie, dass die Datentypen `untypedAtomic`, `dayTimeDuration` und `yearMonthDuration` mit den Candidate Recommendations vom 23. Jänner 2007 aus dem XPath Datentypen-Namespace in den XML-Schema Namespace verschoben

wurden, d.h. `xs:yearMonthDuration`.

Wenn Namespaces für Funktionen, Typ-Konstrukturen, Node Tests usw. falsch zugewiesen wurden, wird ein Fehler ausgegeben. Beachten Sie jedoch, dass einige Funktionen denselben Namen wie Schema-Datentypen haben, z.B. `fn:string` und `fn:boolean`. (Sowohl `xs:string` als auch `xs:boolean` ist definiert.) Das Namespace-Präfix legt fest, ob die Funktion oder der Typ-Konstruktor verwendet wird.

XML-Quelldokument und Validierung
XML-Dokumente, die bei der Ausführung eines XQuery-Dokuments mit dem Altova XQuery 1.0-Prozessor verwendet werden, müssen wohlgeformt sein. Sie müssen jedoch nicht gemäß einem XML-Schema gültig sein. Wenn die Datei nicht gültig ist, wird die ungültige Datei ohne Schemainformationen geladen. Wenn die XML-Datei mit einem externen Schema verknüpft ist und gemäß diesem Schema gültig ist, werden für die XML-Daten nachträglich Validierungsinformationen generiert und für die Überprüfung der Abfrage verwendet.

Statische und dynamische Typ-Überprüfung
In der statischen Analysephase werden Aspekte der Abfrage überprüft wie z.B. die Syntax, ob externe Referenzen (z.B. für Module) vorhanden sind, ob aufgerufene Funktionen und Variablen definiert sind, usw. In der statischen Analysephase wird keine Typ-Überprüfung durchgeführt. Wenn in dieser Phase ein Fehler gefunden wird, wird eine Meldung ausgegeben und die Ausführung wird gestoppt.

Die dynamische Typ-Überprüfung wird in Laufzeit durchgeführt, während die Abfrage ausgeführt wird. Wenn ein Typ mit den Anforderungen einer Operation nicht kompatibel ist, wird ein Fehler ausgegeben. So gibt z.B. der Ausdruck `xs:string("1") + 1` einen Fehler zurück, weil die Operation "Addition" nicht an einem Operanden vom Typ `xs:string` ausgeführt werden kann.

Bibliotheksmodule
Bibliotheksmodule dienen zum Speichern von Funktionen und Variablen, damit diese wiederverwendet werden können. Der Altova XQuery 1.0-Prozessor unterstützt Module, die in einer **einzigen externen XQuery-Datei** gespeichert sind. Eine solche Moduldatei muss im Prolog eine `module`-Deklaration enthalten, in der ein Target Namespace zugewiesen wird. Hier ein Beispielmodul:

```
module namespace libns="urn:module-library";
declare variable $libns:company := "Altova";
declare function libns: webaddress() { "http://www.altova.com" };
```

Alle im Modul deklarierten Funktionen und Variablen gehören zu dem mit dem Modul verknüpften Namespace. Das Modul wird durch Import in eine XQuery-Datei mittels der `import module`-Anweisung im Abfrageprolog verwendet. Die `import module`-Anweisung importiert nur Funktionen und Variablen, die direkt in der Bibliotheksmodul-Datei deklariert sind:

```
import module namespace modlib = "urn:module-library" at
   "modulefilename.xq";
if    ($modlib:company = "Altova")
then   modlib:webaddress()
else   error("No match found.")
```

Externe Funktionen
Externe Funktionen, d.h. diejenigen Funktionen, die das Schlüsselwort `external` verwenden, werden nicht unterstützt:

 © 2007 Altova GmbH

```
declare function hoo($param as xs:integer) as xs:string external;
```

Collations
Die Standard Collation ist die Unicode Codepoint Collation. Derzeit werden keine anderen Collations unterstützt. Vergleiche, wie u.a. die Funktion `fn:max` basieren auf dieser Collation.

Zeichennormalisierung
Es wird keine Zeichennormalisierungsform unterstützt.

Präzision von numerischen Typen

- Der Datentyp `xs:integer` hat eine beliebige Präzision, dh. er kann beliebig viele Stellen haben.
- Der Datentyp `xs:decimal` kann nach dem Dezimalpunkt maximal 20 Stellen haben.
- Die Datentypen `xs:float` und `xs:double` sind auf 15 Stellen beschränkt.

Unterstützung für XQuery-Anweisungen
Die `Pragma` Anweisung wird nicht unterstützt. Gegebenenfalls wird sie ignoriert und der Fallback-Ausdruck wird evaluiert.

Unterstützung für XQuery-Funktionen
Informationen zu implementierungsspezifischem Verhalten von XQuery 1.0-Funktionen finden Sie im Abschnitt XPath 2.0- und XQuery 1.0-Funktionen.

1.4 XPath 2.0- und XQuery 1.0-Funktionen

Die XPath 2.0- und XQuery 1.0-Funktionen werden von folgenden Prozessoren überprüft:

- dem **Altova XPath 2.0-Prozessor**, der (i) eine Komponente des Altova XSLT 2.0-Prozessors ist und (ii) im XPath Evaluator von Altova XMLSpy verwendet wird, um XPath-Ausdrücke hinsichtlich des in XMLSpy aktiven XML-Dokuments auszuwerten.
- dem **Altova XQuery 1.0-Prozessor**.

In diesem Abschnitt wird beschrieben, wie XPath 1.0- und XQuery 1.0-Funktionen vom Altova XPath 2.0-Prozessor und dem Altova XQuery 1.0-Prozessor behandelt werden. Es sind nur die Funktionen aufgelistet, die implementierungsspezifisch behandelt werden oder bei denen sich eine einzelne Funktion in einer der drei Umgebungen, in denen sie verwendet wird (also in XSLT 2.0, in XQuery 1.0 und im XPath Evaluator von XMLSpy), anders verhält. Beachten Sie, dass in diesem Abschnitt nicht beschrieben wird, wie diese Funktionen verwendet werden. Nähere Informationen über die Verwendung dieser Funktionen finden Sie im XQuery 1.0 and XPath 2.0 Functions and Operators PR des W3C vom 23. Jänner 2007.

1.4.1 Allgemeine Informationen

Standardkonformität

- Der Altova XPath 2.0-Prozessor implementiert die XPath 2.0 Recommendation vom 23. Jänner 2007. Der Altova XQuery 1.0-Prozessor implementiert die XQuery 1.0 Recommendation vom 23. Jänner 2007. Die Unterstützung der XPath 2.0- und XQuery 1.0-Funktionen in diesen beiden Prozessoren entspricht den Vorgaben in den XQuery 1.0 and XPath 2.0 Functions and Operators Recommendation vom 23. Jänner 2007.
- Der Altova XPath 2.0-Prozessor entspricht den Vorgaben für XML 1.0 (Fourth Edition) und XML Namespaces (1.0).

Default Functions Namespace
Der Default Functions Namespace wurde dem Standard entsprechend eingestellt. Funktionen können daher ohne Präfix aufgerufen werden.

Boundary-whitespace-only-Nodes in XML-Quelldokumenten
Aus den XML-Daten (und somit dem XML-Infoset), die an den Altova XPath 2.0- und den Altova XQuery 1.0-Prozessor übergeben werden, werden boundary-whitespace-only Text Nodes entfernt. (Ein boundary-whitespace-only Text Node ist ein Child-Textnode, der nur Whitespaces enthält und der zwischen zwei Elementen innerhalb eines mixed-content-Elements vorkommt). Dies kann sich auf den Wert auswirken, der von den Funktionen `fn:position()`, `fn:last()`, `fn:count()` und `fn:deep-equal()` zurückgegeben wird.

In jeder Node-Auswahl, bei der auch Text Nodes ausgewählt werden, sind normalerweise auch Boundary-whitespace-only Text Nodes enthalten. Da jedoch beim XML Infoset, das von den Altova-Prozessoren verwendet wird, boundary-whitespace-only Text Nodes entfernt werden, sind diese Nodes im XML Infoset nicht vorhanden. Folglich unterscheidet sich die Größe der Auswahl und die Nummerierung der Nodes von einer, in der diese Text Nodes enthalten sind. Aus diesem Grund können sich die Ergebnisse der Funktionen `fn:position()`, `fn:last()`, `fn:count()` und `fn:deep-equal()` von denen anderer Prozessoren unterscheiden.

Am häufigsten tritt die Situation, dass boundary-whitespace-only Text Nodes als gleichrangige Elemente anderer Elemente überprüft werden, dann auf, wenn `xsl:apply-templates` verwendet wird, um Templates anzuwenden. Wenn die Funktionen `fn:position()`, `fn:last()` und `fn:count()` in Patterns mit einer Namensüberprüfung verwendet werden

 © 2007 Altova GmbH

(z.B. `para[3]`, kurz für `para[position()=3]`), sind boundary-whitespace-only Nodes irrelevant, da nur die benannten Elemente (`para` im obigen Beispiel) ausgewählt werden. (Beachten Sie jedoch, dass boundary-whitespace-only Nodes in Patterns, in denen ein Platzhalterzeichen wie z.B. `*[10]` verwendet wird, sehr wohl relevant sind).

Numerische Notation
Wenn bei der Ausgabe ein `xs:double` in einen String konvertiert wird, wird die logarithmische Darstellung (zB. `1.0E12`) verwendet, wenn der absolute Wert kleiner als 0,000001 oder größer als 1.000.000 ist. Andernfalls wird der Wert als Dezimalwert oder Integer angegeben.

Präzision von `xs:decimal`
Die Präzision bezieht sich auf die Anzahl der Stellen in einer Zahl. Laut Spezifikation sind mindestens 18 Stellen erforderlich. Bei Divisionen, bei denen ein Ergebnis vom Typ `xs:decimal` erzeugt wird, beträgt die Präzision 19 Kommastellen ohne Runden.

Implizite Zeitzone
Beim Vergleich zweier `date`, `time`, oder `dateTime` Werte muss die Zeitzone der verglichenen Werte bekannt sein. Wenn die Zeitzone in einem solchen Wert nicht explizit angegeben ist, wird die implizite Zeitzone verwendet. Als implizite Zeitzone wird die der Systemuhr verwendet. Der Wert kann mit Hilfe der Funktion `fn:implicit-timezone()` überprüft werden.

Collations
Es wird nur die Unicode Codepoint Collation unterstützt. Es können keine anderen Collations verwendet werden. Der Vergleich von Strings, u.a. auch für die Funktionen `fn:max` und `fn:min` basiert auf dieser Collation.

Namespace-Achse
Die Namespace-Achse wird in XPath 2.0 nicht mehr verwendet, wird aber weiterhin unterstützt. Um Namespace-Informationen mit XPath 2.0-Mechanismen aufzurufen, verwenden Sie die Funktionen `fn:in-scope-prefixes()`, `fn:namespace-uri()` und `fn:namespace-uri-for-prefix()`.

Static Type Extensions
Die optionale Funktion zum Überprüfen von statischen Typen wird nicht unterstützt.

1.4.2 Unterstützung von Funktionen

In der nachfolgenden Tabelle wird das implementierungsspezifische Verhalten bestimmter Funktionen (in alphabetischer Reihenfolge) aufgelistet. Beachten Sie bitte die folgenden allgemeinen Punkte:

- Wenn bei einer Funktion eine Sequenz von einem Datenelement als Argument erwartet wird und eine Sequenz von mehr als einem Datenelement gesendet wird, wird im allgemeinen ein Fehler zurückgegeben.
- Alle String-Vergleiche werden unter Verwendung der Unicode Codepoint Collation ausgeführt.
- Ergebnisse, bei denen es sich um QNames handelt, werden in der Form `[prefix:]localname` serialisiert.

Funktionsname	Anmerkungen
`base-uri`	• Wenn im XML-Quelldokument externe Entities verwendet werden und ein Node in der externen Entity als Input-Node-Argument der Funktion base-uri()definiert ist, wird trotzdem die Basis-URI des inkludierenden XML-Dokuments verwendet und nicht die Basis-URI der externen Entity. • Die Basis-URI eines Node im XML-Dokument kann mit Hilfe des Attributs `xml:base` geändert werden.
`collection`	• Die Funktion `collection()` ist das Mapping einer Form `(string, nodes)`, wird derzeit als `available collections` bezeichnet und wird von der externen Umgebung leer gelassen. Die Funktion gibt daher entweder (i) die leere Sequenz zurück (bei Aufruf ohne ein Argument oder mit einer leeren Sequenz) oder (ii) einen Fehler (bei Aufruf mit einem nicht leeren Argument).
`count`	• Siehe Anmerkungen zu Whitespaces im Abschnitt Allgemeine Informationen.
`current-date, current-dateTime, current-time`	• Das aktuelle Datum und die aktuelle Uhrzeit werden aus der Systemuhr übernommen. • Die Zeitzone wird von der impliziten Zeitzone übernommen, die durch den Auswertungskontext geliefert wird; die implizite Zeitzone stammt von der Systemuhr. • Die Zeitzone wird im Ergebnis immer angegeben.
`deep-equal`	• Siehe Anmerkung zu Whitespace im Abschnitt Allgemeine Informationen.
`doc`	• Es wird nur dann ein Fehler ausgegeben, wenn unter dem angegebenen Pfad keine XML-Datei vorhanden ist oder wenn die Datei nicht wohlgeformt ist. Die Datei wird validiert, wenn ein Schema vorhanden ist. Wenn die Datei nicht gültig ist, so wird die ungültige Datei ohne Schema-Informationen geladen.
`id`	• In einem wohlgeformten aber ungültigen Dokument, das zwei oder mehr Elemente mit demselben ID-Wert enthält, wird das erste Element in der Dokumentenreihenfolge zurückgegeben.
`in-scope-prefixes`	• Im XML-Dokument dürfen nur Default Namespaces entdeklariert werden. Doch selbst wenn ein Default Namespace an einem Element-Node entdeklariert wird, wird das Präfix für den Default Namespace, welches der Nulllängenstring ist, für diesen Node zurückgegeben.
`last`	• Siehe Anmerkung zu Whitespace im Abschnitt Allgemeine Informationen.

© 2007 Altova GmbH

`lower-case`	• Es wird nur der ASCII-Zeichensatz unterstützt.
`normalize-unicode`	• Im XML-Dokument dürfen nur Default Namespaces entdeklariert werden. Doch selbst wenn ein Default Namespace an einem Element-Node entdeklariert wird, wird das Präfix für den Default Namespace, welches der Nulllängenstring ist, für diesen Node zurückgegeben.
`position`	• Siehe Anmerkung zu Whitespace im Abschnitt Allgemeine Informationen.
`resolve-uri`	• Wenn das zweite, optionale Argument weggelassen wird, wird die aufzulösende URI (das erste Argument) gegen die Basis-URI aus dem statischen Kontext aufgelöst. Dies ist die URI des XSLT Stylesheets oder die im Prolog des XQuery-Dokuments angegebene Basis-URI. • Die relative URI (das erste Argument) wird nach dem letzten "/" in der Pfadnotation der Basis-URI-Notation angehängt. • Wenn der Wert des ersten Arguments der Nulllängenstring ist, wird die Basis-URI aus dem statischen Kontext zurückgegeben und diese URI inkludiert den Dateinamen des Dokuments, von dem die Basis-URI des statischen Kontexts abgeleitet wird (z.B. die XSLT- oder XML-Datei).
`static-base-uri`	• Die Basis-URI aus dem statischen Kontext ist die Basis-URI des XSLT Stylesheets oder die im Prolog des XQuery-Dokuments angegebene Basis-URI. • Bei Verwendung des XPath Evaluators in der XMLSpy IDE ist die Basis-URI aus dem statischen Kontext die URI des aktiven XML-Dokuments.
`upper-case`	• Es wird nur der ASCII-Zeichensatz unterstützt.

2 Datentypen in XML-Schemas, die von DB generiert wurden

Bei Generierung eines XML-Schemas anhand einer Datenbank (DB), werden die Datentypen, die für diese DB spezifisch sind, in XML-Schema-Datentypen konvertiert. In diesem Anhang finden Sie Informationen zum Mapping von DB-Datentypen auf XML-Schema-Datentypen für gebräuchliche Datenbanken. Wählen Sie die gewünschte Datenbank aus der Liste aus.

- MS Access
- MS SQL Server
- MySQL
- Oracle
- ODBC
- ADO
- Sybase

© 2007 Altova GmbH

2.1 MS Access

Bei Generierung eines XML-Schemas anhand einer MS Access-Datenbank (DB) werden die MS Access DB-Datentypen wie unten in der Tabelle gezeigt in XML-Schema-Datentypen konvertiert.

MS Access-Datentyp	XML-Schema-Datentyp
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

2.2 MS SQL Server

Bei Generierung eines XML-Schemas anhand einer MS SQL-Datenbank (DB) werden die MS SQL Server DB-Datentypen wie unten in der Tabelle gezeigt in XML-Schema-Datentypen konvertiert.

MS SQL Server-Datentyp	XML-Schema-Datentyp
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

© 2007 Altova GmbH

2.3 MySQL

Bei Generierung eines XML-Schemas anhand einer MySQL-Datenbank (DB) werden die MySQL-Datentypen wie unten in der Tabelle gezeigt in XML-Schema-Datentypen konvertiert.

My SQL-Datentyp	XML-Schema-Datentyp
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

2.4 Oracle

Bei Generierung eines XML-Schemas anhand einer Oracle-Datenbank (DB) werden die Oracle DB-Datentypen wie unten in der Tabelle gezeigt in XML-Schema-Datentypen konvertiert.

Oracle-Datentyp	XML-Schema-Datentyp
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

* Wenn ein Check Constraint auf eine Spalte vom Datentyp `NUMBER` angewendet wird und der Check Constraint auf die Werte `0` oder `1` überprüft, wird der Datentyp `NUMBER` für diese Spalte in einem XML-Schema-Datentyp `xs: boolean`. konvertiert. Dieser Mechanismus eignet sich zum Generieren eines Datentyps `xs: boolean` im generierten XML-Schema.

© 2007 Altova GmbH

2.5 ODBC

Bei Generierung eines XML-Schemas anhand einer ODBC-Datenbank (DB) werden die ODBC-Datentypen wie unten in der Tabelle gezeigt in XML-Schema-Datentypen konvertiert.

ODBC-Datentyp	XML-Schema-Datentyp
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

2.6 ADO

Bei Generierung eines XML-Schemas anhand einer ADO-Datenbank (DB) werden die ADO DB-Datentypen wie unten in der Tabelle gezeigt in XML-Schema-Datentypen konvertiert.

ADO-Datentyp	XML-Schema-Datentyp
adGUID	xs:ID
adChar	xs:string
adWChar	xs:string
adVarChar	xs:string
adWVarChar	xs:string
adLongVarChar	xs:string
adWLongVarChar	xs:string
adVarWChar	xs:string
adBoolean	xs:boolean
adSingle	xs:float
adDouble	xs:double
adNumeric	xs:decimal
adCurrency	xs:decimal
adDBTimeStamp	xs:dateTime
adDate	xs:date
adBinary	xs:base64Binary
adVarBinary	xs:base64Binary
adLongVarBinary	xs:base64Binary
adInteger	xs:integer
adUnsignedInt	xs:unsignedInt
adSmallInt	xs:short
adUnsignedSmallInt	xs:unsignedShort
adBigInt	xs:long
adUnsignedBigInt	xs:unsignedLong
adTinyInt	xs:byte
adUnsignedTinyInt	xs:unsignedByte

© 2007 Altova GmbH

2.7 Sybase

Bei Generierung eines XML-Schemas anhand einer Sybase-Datenbank (DB) werden die Sybase DB-Datentypen wie unten in der Tabelle gezeigt in XML-Schema-Datentypen konvertiert.

Sybase-Datentyp	XML-Schema-Datentyp
char	xs:string
nchar	xs:string
varchar	xs:string
nvarchar	xs:string
text	xs:string
sysname=varchar(30)	xs:string
bit	xs:boolean
real	xs:float
float	xs:float
double	xs:double
decimal	xs:decimal
money	xs:decimal
smallmoney	xs:decimal
datetime	xs:dateTime
smalldatetime	xs:dateTime
timestamp	xs:dateTime
binary<=255	xs:base64Binary
varbinary<=255	xs:base64Binary
image	xs:base64Binary
integer	xs:integer
smallint	xs:short
tinyint	xs:byte

3 Datentypen in DB, die von XML-Schema generiert wurden

Bei Generierung einer DB-Struktur anhand eines XML-Schemas werden die für diese DB spezifischen Datentypen anhand von XML-Schema-Datentypen generiert. Informationen zum Mapping von XML-Schema-Datentypen auf DB-Datentypen bei gebräuchlichen Datenbanken finden Sie in diesem Anhang. Wählen Sie das gewünschte Kapitel aus der nachfolgenden Liste aus.

- MS Access
- MS SQL Server
- MySQL
- Oracle

© 2007 Altova GmbH

3.1 MS Access

Bei Generierung einer MS Access-Datenbank (DB) anhand eines XML-Schemas werden XML-Schema-Datentypen wie unten in der Tabelle gezeigt in die MS Access Datentypen konvertiert.

XML-Schema-Datentyp	MS Access-Datentyp
xs:ID	GUID
xs:string	If no facets varchar(255)
	Size = either length or maxLength
	If Size <= 255 varchar(n)
	else memo
xs:normalizedString	*Same as* xs:string
xs:token	*Same as* xs:string
xs:Name	*Same as* xs:string
xs:NCName	*Same as* xs:string
xs:anyURI	*Same as* xs:string
xs:QName	*Same as* xs:string
xs:NOTATION	*Same as* xs:string
xs:boolean	bit
xs:float	Number(single)
xs:double	Number(double)
xs:decimal	Decimal
xs:duration	Date/Time
xs:dateTime	Date/Time
xs:time	Date/Time
xs:date	Date/Time
xs:gYearMonth	Date/Time
xs:gYear	Date/Time
xs:gMonthDay	Date/Time
xs:gDay	Date/Time
xs:gMonth	Date/Time
xs:hexBinary	If no facets varbinary(255)
	Size = either length or maxLength
	If Size <= 8000 varbinary

	else image (OLE Object)
xs:base64Binary	*Same as* xs:hexBinary
xs:integer	Number (Long Integer)
xs:int	Number (Long Integer)
xs:negativeInteger	Number (Long Integer); value constraint
xs:positiveInteger	Number (Long Integer); value constraint
xs:nonNegativeInteger	Number (Long Integer); value constraint
xs:nonPositiveInteger	Number (Long Integer); value constraint
xs:unsignedInt	Number (Long Integer)
xs:short	–no equivalent–
xs:unsignedShort	–no equivalent–
xs:long	–no equivalent–
xs:unsignedLong	–no equivalent–
xs:byte	Number (Byte)
xs:unsignedByte	Number (Byte)

© 2007 Altova GmbH

3.2 MS SQL Server

Bei Generierung eines XML-Schemas anhand einer MS SQL Server-Datenbank (DB) werden die MS SQL Server-Datentypen wie unten in der Tabelle gezeigt in XML-Schema-Datentypen konvertiert.

XML-Schema-Datentyp	MS SQL Server-Datentyp
ID	uniqueidentifier
xs:string	If no facets
	{ifUNICODE nvarchar(255)
	else varchar(255)}
	else
	{ifUNICODE
	(Size=either length or maxLength)
	If Size <= 4000
	if FacetLength IS Set then nChar
	else nVarChar
	if Size <= 1073741823 then nText}
	else
	{ifNON-UNICODE
	(Size=either length or maxLength)
	If Size <= 8000
	if FacetLength IS Set then char
	else varchar
	if Size <= 2147483647 then text}
xs:normalizedString	*Same as* xs:string
xs:token	*Same as* xs:string
xs:Name	*Same as* xs:string
xs:NCName	*Same as* xs:string
xs:anyURi	*Same as* xs:string
xs:QName	*Same as* xs:string
xs:NOTATION	*Same as* xs:string
xs:boolean	bit
xs:float	real

xs:double	float
xs:decimal	decimal
xs:duration	datetime
xs:dateTime	datetime
xs:time	datetime
xs:date	datetime
xs:gYearMonth	datetime
xs:gYear	datetime
xs:gMonthDay	datetime
xs:gDay	datetime
xs:gMonth	datetime
xs:hexBinary	If no facets varbinary(255)
	(Size = either length or maxLength
	If Size <= 8000
	if FacetLength IS set 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)

© 2007 Altova GmbH

3.3 MySQL

Bei Generierung eines XML-Schemas anhand einer MySQL Datenbank (DB) werden die MySQL DB-Datentypen wie unten in der Tabelle gezeigt in XML-Schema-Datentypen konvertiert.

XML-Schema-Datentyp	MySQL-Datentyp
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 medium text
	else if maxlength is set and <= 429496295 then long 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	tinyint(1)
xs:float	float
xs:double	double
xs:decimal	decimal
xs:duration	timestamp
xs:dateTime	datetime
xs:time	time
xs:date	date
xs:gYearMonth	timestamp(4)
xs:gYear	year(4)
xs:gMonthDay	timestamp(8); constraints to check month, day

xs:gDay	timestamp(8); constraints to check day
xs:gMonth	timestamp(8); constraints to check month
xs:hexBinary	If no facets then blob(255)
	else if facet length is set and <= 255 then blob
	else if facet maxLength is set and <= 255 then tinyblob
	else if maxlength is set and <= 65545 then blob
	else if maxlength is set and <= 16777215 then mediumblob
	else if maxlength is set and <= 429496295 then longblob
xs:base64Binary	*Same as* xs:hexBinary
xs:integer	Integer
xs:int	int
xs:negativeInteger	Integer(constrained to {..,-2,-1})
xs:positiveInteger	Integer(constrained to {1,2,..})
xs:nonNegativeInteger	Integer(constrained to {0,1,2,..})
xs:nonPositiveInteger	Integer(constrained to {..,-2,-1,0})
xs:unsignedInt	Int(additional constraints)
xs:short	SmallInt
xs:unsignedShort	SmallInt(additional constraints)
xs:long	Bigint
xs:unsignedLong	Bigint(additional constraints)
xs:byte	Tinyint
xs:unsignedByte	Tinyint(additional constraints)

© 2007 Altova GmbH

3.4 Oracle

Bei Generierung eines XML-Schemas anhand einer Oracle-Datenbank (DB) werden die Oracle DB-Datentypen wie unten in der Tabelle gezeigt in XML-Schema-Datentypen konvertiert.

XML-Schema-Datentyp	Oracle-Datentyp
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 (constained to {..., -2, -1})
xs:positiveInteger	NUMBER (constained to {1, 2, ...})
xs:nonNegativeInteger	NUMBER (constained to {0, 1, 2, ...})
xs:nonPositiveInteger	NUMBER (constained 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)

© 2007 Altova GmbH

4 Technische Daten

Dieses Kapitel enthält nützliche Hintergrundinformationen zu den technischen Aspekten Ihrer Software. Es ist in die folgenden Abschnitte gegliedert:

- OS- und Arbeitsspeicheranforderungen
- Altova XML Parser
- Altova XSLT- und XQuery-Prozessor
- Unicode-Unterstützung
- Internet-Verwendung

4.1 OS- und Arbeitsspeicheranforderungen

Betriebssystem
Diese Software-Anwendung ist eine 32-Bit Windows-Applikation, die auf Windows 2000 und Windows XP läuft.

Arbeitsspeicher
Da die Software in C++ geschrieben wurde, wird dafür nicht so viel Platz wie in einer Java Runtime Umgebung benötigt und normalerweise wird dafür weniger Arbeitsspeicher als bei einer vergleichbaren Java-basierten Applikation benötigt.
Es ist notwendig, dass die einzelnen Dokumente in den Hauptarbeitsspeicher geladen werden, damit jedes Dokument zur Gänze geparst und analysiert werden kann und die Anzeige- und Bearbeitungsgeschwindigkeit während der normalen Arbeit verbessert wird.

Auch die unbegrenzte Rückgängig-Funktion kann einiges an Arbeitsspeicher in Anspruch nehmen. Wenn Sie große Abschnitte in großen Dokumenten immer wieder ausschneiden und einfügen, kann dies enorm viel Speicherplatz verbrauchen.

© 2007 Altova GmbH

4.2 Altova XML Parser

Wenn Sie ein XML-Dokument öffnen, verwendet die Applikation den integrierten Validierungsparser (den Altova XML Parser), um das Dokument auf Wohlgeformtheit zu prüfen, es gegen ein Schema zu validieren (falls eines angegeben wurde) und Baumstrukturen und Infosets zu erstellen. Der Altova XML Parser dient auch dazu, beim Editieren von Dokumenten intelligente Eingabehilfen zur Verfügung zu stellen und etwaige Validierungsfehler dynamisch anzuzeigen.

Im integrierten Altova XML Parser ist die Final Recommendation der W3C XML Schema Spezifikation implementiert. Neue Entwicklungen, die von der XML Schema-Arbeitsgruppe empfohlen werden, werden ständig in den Altova Parser integriert, sodass Ihnen mit Altova-Produkten immer eine Entwicklungsumgebung auf dem neuesten Stand der Technik zur Verfügung steht.

4.3 Altova XSLT- und XQuery-Prozessor

Die Altova Produkte arbeiten mit dem Altova XSLT 1.0-Prozessor, dem Altova XSLT 2.0-Prozessor und dem Altova XQuery-Prozessor. Nähere Informationen zu diesen Prozessoren finden Sie im Anhang 1 zu dieser Dokumentation im Abschnitt über Prozessoren.

Diese drei Prozessoren stehen auch als eigenständige Applikationen in Form des AltovaXML -Pakets zum kostenlosen Download von der Altova Website zur Verfügung. Die Dokumentation zur Verwendung der Prozessoren wird mit dem AltovaXML-Paket mitgeliefert.

© 2007 Altova GmbH

4.4 Unicode-Unterstützung

Unicode ist ein neuer 16-Bit-Zeichensatzstandard, der vom Unicode-Konsortium definiert wurde und der eine eindeutige Zahl für jedes Zeichen zur Verfügung stellt, und zwar:

- plattformunabhängig
- programmunabhängig
- unabhängig von der Sprache

Im Prinzip arbeiten Computer nur mit Zahlen. Sie speichern Buchstaben und andere Zeichen unter einer Zahl, die sie diesen zuweisen. Vor der Entwicklung von Unicode gab es hunderte von verschiedenen Codiersystemen für die Zuweisung dieser Zahlen. Es gab keine einzige Codierung, die Platz für genügend Zeichen bot: So werden z.B. in der Europäischen Union mehrere verschiedene Codierungen benötigt, um alle ihre Sprachen abzudecken. Selbst für eine einzige Sprache wie z.B. Englisch reichte eine einzelne Codierung nicht aus, um alle gebräuchlichen Buchstaben, Satzzeichen und technischen Symbole abzudecken.

Diese Codierungssysteme standen früher miteinander in Konflikt, d.h. zwei Codierungen verwendeten dieselbe Zahl für zwei verschiedene Zeichen oder verschiedene Zahlen für dasselbe Zeichen. Jeder Computer (v.a. Server) muss viele verschiedene Codierungen unterstützen. Wenn aber Daten zwischen verschiedenen Codierungen oder Plattformen ausgetauscht werden, besteht immer das Risiko, dass Fehler entstehen.

Unicode ist die Lösung für diese Probleme!
Unicode enthält eine eindeutige Zahl für jedes Zeichen, egal für welche Plattform, welches Programm oder welche Sprache. Der Unicode-Standard wurde von branchenführenden Unternehmen wie Apple, HP, IBM, JustSystem, Microsoft, Oracle, SAP, Sun, Base und vielen anderen übernommen.

Unicode ist für moderne Standards wie XML, Java, ECMAScript (JavaScript), LDAP, CORBA 3.0, WML, usw. erforderlich und ist das offizielle Verfahren zum Implementieren von ISO/IEC 10646. Unicode wird von vielen Betriebssystemen, allen modernen Browsern und vielen anderen Produkten unterstützt. Die Entwicklung des Unicode-Standards und die Verfügbarkeit von Tools, die diesen unterstützen, gehören zu den wichtigsten globalen Software Technologie-Trends in letzter Zeit.

Die Integration von Unicode in Client-Server oder Multi-Tier-Applikationen und Websites bietet beträchtliche Kostenvorteile im Vergleich zur Verwendung herkömmlicher Zeichensätze. Dank Unicode kann ein einziges Software-Produkt oder eine einzige Website für mehrere Plattformen, Sprachen und Länder konzipiert werden, ohne dass es neu entwickelt werden muss. Daten können dadurch durch viele verschiedene Systeme übertragen werden, ohne dass dabei Fehler auftreten.

4.4.1 Windows 2000 und Windows XP

Die XML-Produkte von Altova bieten vollständige Unicode-Unterstützung. Um ein XML-Dokument von einem System mit nicht lateinischer Zeichensatzcodierung bearbeiten zu können, benötigen Sie allerdings auch eine Schriftart, die die von diesem Dokument verwendeten Unicode-Zeichen unterstützt.

Beachten Sie bitte, dass die meisten Schriftarten nur eine bestimmte Untergruppe des gesamten Unicode-Bereichs enthalten und normalerweise für das entsprechende Schriftsystem ausgelegt sind. Aus diesem Grund kann es vorkommen, dass ein XML-Dokument "nicht druckbare" Zeichen enthält, da die gewählte Schriftart die erforderlichen Glyphen nicht enthält. Es ist daher manchmal nützlich, wenn man eine Schriftart hat, die den gesamten

Unicode-Bereich abdeckt - v.a. wenn Sie XML-Dokumente aus der ganzen Welt editieren.

Die größte uns bekannte Universalschriftart ist "Arial Unicode MS", eine Schriftart, die von Agfa Monotype für Microsoft entwickelt wurde. Diese Schriftart enthält mehr als 50.000 Glyphen und deckt das gesamte im Unicode 2.1.-Standard definierte Zeichenspektrum ab. Die Schriftart ist in Microsoft Office 2000 enthalten und beansprucht 23 MB.

Wir empfehlen Ihnen dringend, diese Schriftart auf Ihrem System zu installieren und mit der Applikation zu verwenden, wenn Sie öfter Dokumente bearbeiten, die in unterschiedlichen Schriftsystemen verfasst wurden. Diese Schriftart wird bei Verwendung der Standardinstallation von Microsoft Office nicht installiert, Sie können jedoch die Option "Benutzerdefinierte Installation" wählen, um sie zu installieren.

Im Ordner "Examples" finden Sie außerdem eine neue XHTML-Datei mit dem Namen `Unicode-UTF8.html`, die den Satz "Wenn die Welt miteinander spricht, spricht sie Unicode" in vielen verschiedenen Sprachen und Schriftsystemen enthält ("When the world wants to talk, it speaks Unicode") (世界的に話すなら、Unicode です。) - Diese Zeile stammt aus der 10. Unicode-Konferenz von 1997 und ist ein schönes Beispiel für die Bedeutung von Unicode für den XML-Standard. Wenn Sie diese Datei öffnen, erhalten Sie einen kurzen Eindruck davon, was mit Unicode möglich ist und welche Schriftsysteme von den auf Ihrem PC verfügbaren Schriftarten unterstützt werden.

4.4.2 Rechts nach Links-Schreibrichtung

Beachten Sie bitte, dass selbst unter Windows NT 4.0 Texte aus Schriftsystemen mit einer Schreibrichtung von rechts nach links (z.B. Hebräisch oder Arabisch) nur in den Ländern korrekt wiedergegeben werden, in denen solche Schriftsysteme tatsächlich verwendet werden. Dies liegt daran, dass nur die hebräische und arabische Version von Windows NT die Wiedergabe und Bearbeitung von Text in Rechts-links-Schreibweise auf Betriebssystemebene unterstützen.

© 2007 Altova GmbH

4.5 Internet-Verwendung

Altova-Applikationen können für Sie auch eine Verbindung mit dem Internet herstellen. Dies geschieht in den folgenden Fällen:

- Wenn Sie im Registrierungsdialogfeld (**Hilfe | Software-Aktivierung**) auf "Kostenlosen Evaluierungs-Key anfordern" klicken, werden die drei Felder im Registrierungsdialogfeld über eine normale HTTP-Verbindung (Port 80) an unseren Webserver übertragen. Sie erhalten dann per E-Mail (normales SMTP) den kostenlosen Evaluierungs-Keycode zugesandt.
- Falls Sie das Dialogfeld URL öffnen verwenden, um ein Dokument direkt über eine URL zu öffnen, wird dieses Dokument über eine HTTP-Verbindung (Port 80) aufgerufen. (*Diese Funktionalität steht in XMLSpy und Authentic Desktop zur Verfügung.)*
- Wenn Sie ein XML-Dokument öffnen, das sich auf ein XML-Schema oder eine DTD bezieht und das Dokument durch eine URL definiert wird, wird es ebenfalls über eine HTTP-Verbindung (Port 80) aufgerufen, sobald Sie das XML-Dokument validieren. Dies kann auch automatisch beim Öffnen eines Dokuments vorkommen, wenn Sie die Applikation so konfiguriert haben, dass Dateien beim Öffnen automatisch validiert werden (Register "Datei" des Dialogfelds **Extras | Optionen**).
- Wenn Sie den Befehl "Als Mail senden..." (**Datei | Als Mail senden**) verwenden, wird die aktuelle Auswahl bzw. Datei über ein MAPI-kompatibles Mail-Programm, das auf dem PC des Benutzers installiert ist, versendet.

5 Lizenzinformationen

Dieser Anhang enthält die folgenden Informationen:

- Informationen über den Vertrieb dieses Software-Produkts
- Copyright-Informationen im Zusammenhang mit diesem Software-Produkt
- Die Endbenutzer-Lizenzvereinbarung zu diesem Software-Produkt

Lesen Sie die Informationen bitte sorgfältig - sie sind rechtlich bindend, da Sie sich bei der Installation dieses Software-Produkts damit einverstanden erklärt haben.

© 2007 Altova GmbH

5.1 Electronic Software Distribution

Dieses Produkt ist über EDS (Electronic Software Distribution), also auf elektronischem Weg erhältlich, eine Methode, die die folgenden einzigartigen Vorteile bietet:

- Sie können die Software kostenlos testen, bevor Sie sich zu einem Kauf entscheiden.
- Wenn Sie sich entschieden haben, die Software zu kaufen, können Sie Ihre Bestellung online abgeben und das Produkt steht Ihnen in Minutenschnelle im vollen Lizenzumfang zur Verfügung.
- Sie erhalten immer die neueste Version unserer Software
- Die Software enthält ein umfassendes Online-Hilfesystem. Die neueste Version des Benutzerhandbuchs steht auf unserer Website www.altova.com (i) im HTML-Format zum Aufrufen online und (ii) im PDF-Format zum Download und Ausdrucken zur Verfügung.

30-Tage-Evaluierungszeitraum
Nachdem Sie dieses Software-Produkt heruntergeladen haben, können Sie es 30 Tage lang kostenlos testen. Während dieses Zeitraums werden Sie nach etwa 20 Tagen in regelmäßigen Abständen daran erinnert, dass die Software noch nicht lizenziert wurde. Diese Erinnerungsmeldung wird allerdings nur einmal, nämlich bei jedem Start des Programms, angezeigt. Wenn Sie das Programm nach Ablauf des 30-tägigen Evaluierungszeitraums weiterverwenden möchten, müssen Sie eine Software-Produktlizenz erwerben, die Sie in Form eines Keycodes erhalten, der in das Dialogfeld "Software-Aktivierung" eingegeben werden muss, um das Produkt zu aktivieren. Sie können Ihre Lizenz online in unserem Web-Shop unter Altova Shop erwerben.

Verteilung des Produkts
Wenn Sie das Produkt gemeinsam mit anderen Usern verwenden wollen, stellen Sie sicher, dass Sie immer nur das Installationsprogramm weitergeben. Es enthält die Applikation, die Beispieldateien und die Online-Hilfe. Jeder, der das Produkt von Ihnen erhält, kann die Software 30 Tage lang kostenlos testen. Nach Ablauf des Testzeitraums, muss eine Lizenz erworben werden, damit das Produkt weiter verwendet werden kann.

Nähere Informationen finden Sie in der Endbenutzer-Lizenzvereinbarung am Ende dieses Handbuchs.

5.2 Lizenzüberwachung

Ihr Altova-Produkt verfügt über ein integriertes Lizenzüberwachungsmodul, das Ihnen hilft, unbeabsichtigte Verletzungen der Lizenzvereinbarung zu vermeiden. Ihr Produkt kann entweder mit einer Einzelplatzlizenz oder einer Mehrfachlizenz erworben werden. Je nach Lizenz stellt das Lizenzüberwachungsmodul sicher, dass nicht mehr als die lizenzierte Anzahl an Benutzern die Applikation gleichzeitig verwendet.

Bei dieser Lizenzüberwachungsmethode wird Ihr LAN-Netzwerk verwendet, um die Kommunikation zwischen Instanzen der Applikation, die auf verschiedenen Computern laufen, zu überwachen.

Einzelplatzlizenz
Beim Start der Applikation wird ein kurzes Datagram abgesendet, um andere Instanzen des Produkts, die auf anderen Computern im selben Netzwerk laufen, zu finden. Wenn keine Antwort einlangt, wird ein Port geöffnet, der Informationen von anderen Instanzen der Applikation empfangen kann. Mehr geschieht nicht bei einer Einzelplatzlizenz. Wenn Sie nicht mit einem LAN verbunden sind oder für Ihre Internetverbindung eine Einwahlverbindung verwenden, verursacht die Applikation keinerlei Netzwerkverkehr.

Mehrfachlizenz
Wenn Sie im selben LAN mehrere Instanzen der Applikation verwenden, kommunizieren diese beim Start kurz miteinander, um Keycode-Informationen auszutauschen um sicherzustellen, dass nicht mehr als die lizenzierte Anzahl an Lizenzen gleichzeitig in Verwendung ist. Dieselbe Lizenzüberwachungstechnologie wird auch bei Unix und vielen anderen Datenbankentwicklungstools verwendet. Sie gestattet Benutzern den Erwerb von Parallellizenzen für mehrere Benutzer zu vernünftigen Preisen.

Beachten Sie, dass Ihr Altova-Produkt niemals versucht, Informationen aus Ihrem LAN oder über das Internet zu versenden. Wir sind außerdem bestrebt, nur wenige, kleine Netzwerkpakete zu versenden, um Ihr Netzwerk nicht zu überlasten. Die von Ihrem Altova Produkt verwendeten TCP/IP Ports (2799) sind offiziell bei IANA registriert, (nähere Informationen siehe http://www.isi.edu/in-notes/iana/assignments/port-numbers) und unser Lizenzüberwachungsmodul basiert auf einer bewährten und getesteten Technologie.

Wenn Sie eine Firewall verwenden, werden Sie unter Umständen feststellen, dass die Computer, auf denen Altova-Produkte laufen, über Port 2799 miteinander kommunizieren. Sie können diesen Netzwerkverkehr zwischen verschiedenen Gruppen in Ihrem Unternehmen natürlich blockieren, solange Sie mit anderen Mitteln sicherstellen können, dass Ihre Lizenzvereinbarung eingehalten wird.

Sie werden außerdem auch feststellen, dass Ihr Altova Produkt viele nützliche Funktionen enthält, die Ihre Internet-Verbindung verwenden, diese stehen jedoch nicht mit der Lizenzüberwachungsfunktion in Zusammenhang.

© 2007 Altova GmbH

5.3 Copyright

All title and copyrights in this software product (including but not limited to images, photographs, animations, video, audio, music, text, and applets incorporated in the product), in the accompanying printed materials, and in any copies of these printed materials are owned by Altova GmbH or the respective supplier. This software product is protected by copyright laws and international treaty provisions.

- This software product ©1998-2007 Altova GmbH. All rights reserved.
- The Sentry Spelling-Checker Engine © 2000 Wintertree Software Inc.
- STLport © 1999, 2000 Boris Fomitchev, © 1994 Hewlett-Packard Company, © 1996, 1997 Silicon Graphics Computer Systems, Inc, © 1997 Moscow Center for SPARC Technology.
- Scintilla © 1998–2002 Neil Hodgson <neilh@scintilla.org>.
- "ANTLR Copyright © 1989-2005 by Terence Parr (www.antlr.org)"

All other names or trademarks are the property of their respective owners.

5.4 Altova Endbenutzer-Lizenzvereinbarung

DIES IST EIN RECHTSGÜLTIGES DOKUMENT -- BITTE BEWAHREN SIE ES SORGFÄLTIG AUF.

ALTOVA® ENDBENUTZER-LIZENZVERTRAG

Lizenzgeber:

Altova GmbH
Rudolfsplatz 13a/9
A-1010 Wien
Österreich

Wichtig – Bitte sorgfältig lesen. Benutzerinformation:

Dieser Endbenutzer-Lizenzvertrag („Lizenzvertrag“) ist ein rechtsgültiger Vertrag, der zwischen Ihnen und Altova GmbH („Altova“) abgeschlossen wurde. Bevor Sie die von Altova bereitgestellte Software („Software“) sowie die dazugehörige Dokumentation (u.a. Unterlagen in gedruckter Form, Online-Dateien oder Dokumentation in elektronischer Form („Dokumentation“)) verwenden, lesen Sie bitte dieses Dokument sorgfältig durch. Indem Sie auf die untenstehenden Schaltflächen „Ich akzeptiere“ und „Weiter“ klicken, oder indem Sie die Software installieren oder auf sonstige Weise nutzen, stimmen Sie zu, dass Sie die Bedingungen dieses Lizenzvertrags sowie die Datenschutzbestimmungen („Datenschutzbestimmungen“) von Altova, inklusive jedoch nicht ausschließlich die nachfolgend festgelegten Garantieausschlüsse, Haftungsbeschränkungen sowie Datennutzungs- und Kündigungsregelungen als rechtsverbindlich anerkennen, auch wenn Sie die Software nicht käuflich erwerben. Sie stimmen zu, dass diese Vereinbarung in demselben Umfang gültig und durchsetzbar ist, wie ein von Ihnen persönlich unterzeichneter Vertrag. Wenn Sie den Bedingungen dieses Lizenzvertrags nicht zustimmen, sind Sie nicht zur Verwendung der Software berechtigt und müssen alle heruntergeladenen Kopien, die sich in Ihrem Besitz oder unter Ihrer Kontrolle befinden, vernichten. Bitte besuchen Sie unsere Website unter http://www.altova.com/eula, um eine Kopie dieses Lizenzvertrags herunterzuladen und auszudrucken. Die Datenschutzbestimmungen können unter http://www.altova.com/privacy eingesehen werden.

1. SOFTWARELIZENZ

(a) **Lizenzgewährung.** Sofern Sie diesem Lizenzvertrag zugestimmt haben, gewährt Ihnen Altova eine nicht ausschließliche und nicht übertragbare (mit Ausnahme der nachfolgenden Bestimmungen) Lizenz zur Installation und Verwendung der Software auf Ihrem kompatiblen Computer, wobei die maximale Zahl an Computern, die im Lizenzumfang enthalten sind, nicht überschritten werden darf. Diese maximale Zahl an Computern wird beim Kauf der Software festgelegt. Während des Evaluierungszeitraums (wie nachfolgend definiert) darf die Software nur von einem Benutzer und auf nur einem Computer installiert und genutzt werden. Wenn Sie die Lizenz im Rahmen eines Pakets von Altova-Softwareprodukten („Produktpaket“) erworben haben, von denen Sie nicht jedes Produkt einzeln installiert haben, gelten die Bestimmungen dieses Lizenzvertrags für die Verwendung der gesamten Software, die im Lieferumfang des Produktpakets enthalten ist. Wenn Sie eine Lizenz für SchemaAgent erworben haben, dann gelten die Bestimmungen dieses Lizenzvertrags zur Nutzung der Software für den darin inkludierten SchemaAgent-Server („SchemaAgent-Server“). Dabei dürfen Sie den

SchemaAgent-Server jedoch ausschließlich in Verbindung mit der Altova-Software und nur für die in der dazugehörigen Dokumentation beschriebenen Zwecke verwenden. Haben Sie außerdem eine Lizenz für XMLSpy Enterprise Edition, MapForce Enterprise Edition oder UModel erworben, dann sind Sie durch die Ihnen im Rahmen dieser Lizenz erteilten Genehmigung zur Installation und Verwendung der Software auch berechtigt, Quellcode zu generieren, und zwar entweder (i) anhand von Modulen der Altova Library, die in der Software inkludiert sind (der auf diese Weise generierte Code wird als „beschränkter Quellcode" bezeichnet), oder (ii) anhand Ihrer eigenen Schemata oder Mappings (der anhand Ihres eigenen Quellmaterials generierte Code wird als „unbeschränkter Quellcode" bezeichnet). Zusätzlich zu den durch diesen Lizenzvertrag eingeräumten Rechten gewährt Ihnen Altova eine nicht ausschließliche, nicht übertragbare und eingeschränkte Lizenz, den gesamten generierten Code (beschränkter und unbeschränkter Quellcode) in eine ausführbare Form zu übersetzen und diese zu verwenden, zu kopieren, weiterzugeben oder zu lizenzieren. Sie sind nur dann befugt, den beschränkten Quellcode an Dritte ab- oder weiterzugeben, weiterzulizenzieren, zu verkaufen oder zu übertragen, wenn die betreffende Partei bereits in Form eines gesonderten Lizenzvertrags oder einer anderen Vereinbarung mit Altova zur Verwendung des beschränkten Quellcodes berechtigt ist. Altova behält sich alle übrigen Rechte an der Software vor. Bei den Bestandteilen von UModel, für die ein Reverse Engineering Ihres eigenen oder eines anderen Quellcodes, den Sie rechtmäßig erworben haben, zulässig ist, stellt eine solche Verwendung durch Sie keinen Verstoß gegen diesen Lizenzvertrag dar. Soweit dies nicht in Abschnitt 1(h) ausdrücklich gestattet wird, ist ein Reverse Engineering der Software explizit untersagt.

(b) **Verwendung auf einem Server.** Sie sind berechtigt, ein Exemplar der Software auf einem Dateiserver in Ihrem Intranet zu installieren und von dort auf die maximale Zahl von Computern innerhalb dieses Netzwerks, die in der Lizenz angegeben ist, herunterzuladen und zu installieren. Wenn Sie eine Lizenz für SchemaAgent erworben haben, können Sie SchemaAgent auf jedem Server oder jeder Workstation installieren und in Verbindung mit Ihrer Software verwenden. Jede sonstige Netzwerknutzung ist unzulässig; dies umfasst insbesondere die direkte Verwendung oder die Nutzung der Software über Befehle, Daten oder Anweisungen von oder an einen Computer, der nicht Teil Ihres internen Netzwerks ist, die Verwendung für Internet- oder Webhosting-Dienste sowie die Verwendung durch einen Benutzer, der nicht unter einer gültigen Lizenz von Altova zur Verwendung dieses Exemplars der Software berechtigt ist. Wenn Sie eine Parallellizenz im Sinne von Abschnitt 1 (c) erworben haben, sind Sie berechtigt, ein Exemplar der Software auf einem Terminalserver in Ihrem Intranet zu installieren, damit einzelne Benutzer in Ihrer Organisation von einem anderen Computer innerhalb des Netzwerks auf die Software zugreifen und diese nutzen können. Jedoch darf die Zahl der Benutzer, die über einen Netzwerk- oder Terminalserver auf die Software zugreifen, die maximale Zahl der im Lizenzumfang enthaltenen User nicht überschreiten. Altova gibt über die Performance der Altova-Software in einer Terminalserver-Umgebung keinerlei Zusicherungen oder Gewährleistungen ab. Die Anwendung der Altova-Software auf die vorstehend beschriebene Weise ist ausdrücklich von der eingeschränkten Garantie, die durch Abschnitt 5 dieses Lizenzvertrags eingeräumt wird, ausgeschlossen. Ebenso wird kein technischer Support bei Problemen geleistet, die aufgrund der Anwendung in einer solchen Umgebung auftreten.

(c) **Parallellizenz.** Wenn Sie eine Software-Version für Parallelbenutzer erworben haben, können Sie die Software auf mehreren kompatiblen Computern installieren, jedoch höchstens zehnmal (10-mal) die maximale Zahl der Benutzer, vorausgesetzt, dass nur die maximale Zahl von Benutzern die Software wirklich gleichzeitig nutzen. Die maximale Zahl der Parallelbenutzer wird beim Kauf der

Software festgelegt.

(d) **Kopien für Sicherungs- und Archivierungszwecke.** Sie dürfen eine Kopie der Software für Sicherungs- und eine Kopie für Archivierungszwecke herstellen, sofern diese nicht auf einem Computer installiert oder verwendet werden. Außerdem müssen bei Kopien zu Sicherungs- und Archivierungszwecken alle im Original enthaltenen urheber- und patentrechtlichen Angaben sowie alle anderen Angaben hinsichtlich geistiger Eigentumsrechte von Ihnen reproduziert werden und in der jeweiligen Kopie der Software unverändert enthalten sein. Die Befugnis zum Anlegen einer Sicherungs- und Archivierungskopie darf nur an Dritte übertragen werden, wenn gleichzeitig eine Übertragung aller Rechte an der Software gemäß Abschnitt 3 erfolgt.

(e) **Nutzung auf dem Privat-PC.** Als vorrangiger Benutzer des Computers, auf dem die Software installiert wurde, sind Sie berechtigt, ein weiteres Exemplar der Software zum persönlichen Gebrauch auf einem Ihrer Heimcomputer zu installieren. Die Software darf jedoch nicht gleichzeitig auf Ihrem Heimcomputer und dem vorrangigen Computer verwendet werden.

(f) **Freischaltcodes, Upgrades und Updates.** Bevor Sie die Software kaufen bzw. im Rahmen Ihrer Registrierung für den Evaluierungszeitraum von dreißig (30) Tagen erhalten Sie einen Evaluierungs-Freischaltcode. Wenn Sie sich in der Folge dazu entscheiden, die Software von Altova GmbH oder einem autorisierten Vertriebshändler zu kaufen, erhalten Sie einen uneingeschränkten Freischaltcode. Mit diesem können Sie die Software nach dem Evaluierungszeitraum aktivieren. Ohne ausdrückliche schriftliche Genehmigung von Altova dürfen Sie an Freischaltcodes keine Lizenzen einräumen und Freischaltcodes nicht reproduzieren oder vertreiben. Wenn es sich bei der Software, für die Sie eine Lizenz erworben haben, um ein Upgrade oder ein Update handelt, so ersetzt dieses, je nach Umfang des Upgrades oder Updates, die frühere Version der lizenzierten Software zur Gänze oder teilweise. Durch das betreffende Upgrade oder Update und den dazugehörigen Freischaltcode wird keine zweite Lizenz für die Software gewährt, und Sie dürfen das Upgrade oder Update nicht zusätzlich zu der Software verwenden, die dadurch ersetzt wird. Sie erklären, dass durch die Verwendung eines Upgrades oder Updates Ihr Recht zur Verwendung einer früheren Version der Software (oder eines Teils der Software) erlischt.

(g) **Eigentumsrecht.** Das Eigentumsrecht an der Software wird Ihnen nicht übertragen. Alle Exemplare der Software und alle von Ihnen angefertigte Kopien verbleiben – vorbehaltlich der Ihnen durch diesen Lizenzvertrag eingeräumten Rechte – im Eigentum von Altova. Dokumente, Dateien, Stylesheets, generierter Programmcode (dies beinhaltet auch den unbeschränkten Quellcode) und Schemata, die Sie unter Verwendung der Software anlegen oder erstellen, sind in Übereinstimmung mit der dazugehörigen Dokumentation und den Bestimmungen dieses Lizenzvertrags Ihr Eigentum.

(h) **Reverse Engineering.** Sofern dies nicht ausdrücklich nach geltendem Recht der Europäischen Union gestattet ist, sind Sie nicht berechtigt, an der Software Reverse Engineering durchzuführen, die Software zu dekompilieren, zu disassemblieren, oder auf andere Weise zu versuchen, den Quellcode, zugrunde liegende Konzepte, Techniken für Benutzerschnittstellen oder Algorithmen der Software direkt oder indirekt zu entschlüsseln. Davon ausgenommen ist lediglich das ausdrücklich nach geltendem Recht der Europäischen Union gestattete Maß an Dekompilierung, das erforderlich ist, um die Funktionsfähigkeit der Software mit anderen Softwareprogrammen zu erzielen. Dies ist Ihnen jedoch erst dann gestattet, wenn Sie zuerst bei Altova die dafür notwendigen Informationen angefordert haben und Altova Ihnen diese nicht zur Verfügung gestellt hat. Altova ist berechtigt, für die Bereitstellung solcher Informationen angemessene Bedingungen aufzustellen und eine angemessene Entschädigung zu

 © 2007 Altova GmbH

verlangen. Die von Altova zur Verfügung gestellten Informationen bzw. alle von Ihnen im Sinne der hier festgelegten Bestimmungen rechtmäßig erworbenen Informationen dürfen nur zu den hierin angeführten Zwecken verwendet und keinesfalls an Dritte weitergegeben oder zur Entwicklung von Software genutzt werden, die der Software, die Gegenstand dieses Lizenzvertrags ist, in wesentlichen Aspekten ähnlich ist. Informationen im Sinne dieses Absatzes können von Benutzern aus der Europäischen Union bei der Kundendienstabteilung von Altova angefordert werden.

(i) **Sonstige Beschränkungen.** Sie sind nicht berechtigt, die Software (zur Gänze oder teilweise) zu verleihen, zu vermieten, per Leasing zur Verfügung zu stellen, weiterzulizenzieren, weiterzugeben oder auf sonstige Weise Dritten zu überlassen, es sei denn, dies ist durch die Bestimmungen in Abschnitt 3 oder an anderer Stelle in diesem Lizenzvertrag ausdrücklich gestattet. Sofern dies nicht ausdrücklich in diesem Vertrag gestattet ist, dürfen Sie die Software nicht kopieren. Alle Kopien, zu deren Anfertigung Sie berechtigt sind, müssen die im Original enthaltenen urheber- und patentrechtlichen Angaben sowie alle anderen Angaben hinsichtlich geistiger Eigentumsrechte unverändert enthalten. Die Software darf nicht verändert, adaptiert oder übersetzt werden. Sie dürfen weder direkt noch indirekt ein Pfand- oder Sicherungsrecht an der Software bestellen oder die Bestellung eines solchen zulassen. Es ist Ihnen nicht gestattet, wissentlich eine Handlung vorzunehmen, durch die die Software öffentlich zugänglich wird, oder die Software in einer Computerumgebung zu verwenden, die nicht in diesem Lizenzvertrag angegeben ist. Sie haben die geltenden gesetzlichen Bestimmungen und die Anweisungen von Altova in Bezug auf die Benutzung der Software einzuhalten. Sie sind verpflichtet, Ihre Mitarbeiter und Vertreter, die Zugriff auf die Software haben, von den in diesem Lizenzvertrag enthaltenen Beschränkungen in Kenntnis zu setzen und diese auf sie zu überbinden. SIE TRAGEN DIE ALLEINIGE VERANTWORTUNG FÜR DIE EIGNUNG UND ZWECKMÄSSIGKEIT DER SOFTWARE FÜR DEN BEABSICHTIGTEN VERWENDUNGSZWECK UND WERDEN ALTOVA FÜR JEGLICHE ANSPRÜCHE UND FORDERUNGEN DRITTER, DIE AUF DIE EIGNUNG UND ZWECKMÄSSIGKEIT DER VON IHNEN VERWENDETEN SOFTWARE BEGRÜNDET SIND, SCHAD- UND KLAGLOS HALTEN. DIE SOFTWARE IST KEINESFALLS FÜR DIE VERWENDUNG IN KERNKRAFTANLAGEN, FLUGZEUGNAVIGATIONS-, KOMMUNIKATIONS- ODER LUFTVERKEHRSKONTROLLSYSTEMEN KONZIPIERT, BEI DENEN DER AUSFALL DER SOFTWARE ZU TODESFÄLLEN, PERSONENSCHÄDEN ODER SCHWEREN SACH- ODER UMWELTSCHÄDEN FÜHREN KÖNNTE.

2. GEISTIGE EIGENTUMSRECHTE

Anerkennung der Rechte von Altova. Die Software und alle Kopien, deren Anfertigung Ihnen von Altova gestattet ist, sind das geistige Eigentum von Altova und dessen Lieferanten und stehen in deren Besitz. Struktur, Organisation und Code der Software stellen wertvolle Betriebsgeheimnisse und vertrauliche Informationen von Altova und dessen Lieferanten dar. Die Software ist durch gesetzliche Bestimmungen urheberrechtlich geschützt. Diese gesetzlichen Bestimmungen beinhalten (ohne darauf beschränkt zu sein) das Urheberrecht der Vereinigten Staaten, internationale Verträge und das in den Ländern, in denen die Software genutzt wird, geltende Recht. Sie anerkennen, dass sich Altova das Eigentum an allen Patenten, Urheberrechten, Branchengeheimnissen, Warenzeichen und sonstigen geistigen Eigentumsrechten, die hinsichtlich der Software bestehen, vorbehält. Das Eigentumsrecht von Altova erstreckt sich auch auf alle Bilder, Fotografien, Animationen, Videos, Audioaufzeichnungen, Musikstücke, Texte und „Applets“, die Teil der Software sind, und alle dazugehörigen Unterlagen in gedruckter Form. Sie dürfen keine Handlungen vornehmen, die sich nachteilig auf die geistigen Eigentumsrechte von Altova an der Software auswirken. Warenzeichen dürfen nur in Übereinstimmung

mit den anerkannten Standards für die Verwendung von Warenzeichen (einschließlich der namentlichen Nennung der Warenzeicheninhaber) verwendet werden. Die Verwendung von Warenzeichen ist nur zur Kennzeichnung von Druckmaterialien, die mit der Software hergestellt wurden, gestattet. Es entstehen Ihnen daraus keinerlei Eigentumsrechte an dem betreffenden Warenzeichen. XMLSpy, Authentic, StyleVision, MapForce, Markup Your Mind, Axad, Nanonull und Altova sind (in mehreren Ländern eingetragene) Warenzeichen von Altova GmbH. Unicode und das Logo von Unicode sind Warenzeichen von Unicode, Inc. Windows, Windows 95, Windows 98, Windows NT, Windows 2000 und Windows XP sind Warenzeichen von Microsoft. W3C, CSS, DOM, MathML, RDF, XHTML, XML und XSL sind (in mehreren Ländern eingetragene) Warenzeichen des World Wide Web Consortium (W3C). Marken des W3C sind von den Dachinstitutionen des Konsortiums (MIT, INRIA und Keio) eingetragen und stehen in deren Eigentum. Sofern dies nicht ausdrücklich hierin festgelegt ist, entstehen Ihnen aus diesem Lizenzvertrag keinerlei geistigen Eigentumsrechte an der Software. Mitteilungen über geltend gemachte Verstöße gegen geistige Eigentumsrechte sind an den Copyright Agent von Altova zu richten (nähere Angaben dazu finden Sie auf der Website von Altova).

3. ÜBERTRAGUNGSBESCHRÄNKUNG

Ungeachtet der vorstehenden Bestimmungen dürfen Sie alle Ihre Rechte zur Verwendung der Software an eine andere natürliche oder juristische Person unter der Voraussetzung übertragen, dass: (a) Sie den vorliegenden Lizenzvertrag, die Software und sonstige Soft- oder Hardware, die mit der Software, die Gegenstand dieses Lizenzvertrags ist, geliefert, verpackt oder auf dieser vorinstalliert ist, einschließlich aller Kopien, Updates und früherer Versionen sowie aller Exemplare von Fontsoftware, die in andere Formate konvertiert wurde, an die betreffende Person übertragen; (b) Sie keine Kopien, einschließlich Sicherungskopien und sonstiger, auf einem Computer gespeicherter Kopien, zurückbehalten; (c) der Empfänger von Altova einen persönlichen Freischaltcode von Altova erhält; und (d) der Empfänger die Bestimmungen dieses Lizenzvertrags sowie alle anderen Bestimmungen akzeptiert, zu denen Sie die Softwarelizenz rechtmäßig erworben haben. Ungeachtet der vorstehenden Ausführungen dürfen Sie keine Schulungs-, Vorab- oder Musterkopien der Software an Dritte übertragen.

4. ZUSÄTZLICHE BEDINGUNGEN FÜR VORAB- UND TESTVERSIONEN

Wenn es sich bei dem von Ihnen mit dieser Lizenz erworbenen Produkt um eine unverkäufliche Vorabversion oder um Beta-Software („Pre-Release-Software“) handelt, gelten die in diesem Abschnitt enthaltenen Bedingungen. Diese gelten außerdem für alle Test- und/oder Demoversionen von Altova-Software („Testsoftware“) und bleiben so lange gültig, bis Sie eine Lizenz erwerben. Wenn Bestimmungen dieses Abschnitts im Widerspruch zu anderen, in diesem Lizenzvertrag enthaltenen Bestimmungen stehen, dann gehen die Bedingungen dieses Abschnitts hinsichtlich Pre-Release-Software und Testsoftware allen anderen Bestimmungen dieses Lizenzvertrags vor, soweit dies für die Auflösung des Widerspruchs erforderlich ist. Sie erkennen an, dass es sich bei der Pre-Release-Software um eine Vorabversion handelt, die nicht das endgültige Produkt von Altova darstellt, und in der Fehler, Funktionsstörungen oder andere Probleme auftreten können, die möglicherweise zu einem System- oder Hardwareabsturz oder -fehler bzw. zu Datenverlust führen. DIE PRE-RELEASE-SOFTWARE UND/ODER TESTSOFTWARE WIRD IHNEN DAHER **OHNE GEWÄHR FÜR DEREN VERWENDUNG ODER LEISTUNGSMERKMALE** ZUR VERFÜGUNG GESTELLT, UND ALTOVA ÜBERNIMMT DIESBEZÜGLICH KEINERLEI AUSDRÜCKLICHEN ODER

STILLSCHWEIGENDEN GEWÄHRLEISTUNGS- ODER HAFTUNGSVERPFLICHTUNGEN. SOFERN UND SOWEIT FÜR PRE-RELEASE- UND/ODER TESTSOFTWARE KEIN HAFTUNGSSAUSSCHLUSS, SONDERN NUR EINE HAFTUNGSBESCHRÄNKUNG GESETZLICH ZULÄSSIG IST, BESCHRÄNKT SICH DIE HAFTUNG VON ALTOVA UND DESSEN LIEFERANTEN AUF INSGESAMT FÜNFZIG US-DOLLAR (50 USD). Wenn die Testversion mit einer Time-out-Funktion ausgestattet ist, kann sie nach Ablauf des Evaluierungszeitraums nicht mehr verwendet werden. Mit dem Ablauf dieses Zeitraums endet Ihre Lizenz automatisch, wenn Sie sie nicht verlängern. Der Zugriff auf Dateien, die mit der Testsoftware erstellt wurden, erfolgt auf Ihr alleiniges Risiko. Sie bestätigen, dass Altova Ihnen weder zugesichert noch garantiert hat, die Pre-Release-Software zu einem späteren Zeitpunkt anderen Personen anzukündigen oder zur Verfügung zu stellen. Sie bestätigen ferner, dass Altova sich Ihnen gegenüber weder ausdrücklich noch stillschweigend verpflichtet hat, die Pre-Release-Software öffentlich anzukündigen oder einzuführen, und dass Altova kein Produkt anbieten muss, das der Pre-Release-Software ähnlich oder mit ihr kompatibel ist. Sie bestätigen folglich, dass jegliche Forschungs- oder Entwicklungsarbeit, die Sie in Bezug auf die Pre-Release-Software oder damit verbundene Produkte ausführen, in Ihr alleiniges Risiko fällt. Wenn Altova Sie dazu auffordert, werden Sie für die Dauer dieses Lizenzvertrags Altova Rückmeldung bezüglich Ihrer Tests und der Pre-Release-Software geben, einschließlich entsprechender Berichte über Fehler und Funktionsstörungen. Wenn Sie die Pre-Release-Software im Rahmen einer gesonderten schriftlichen Vereinbarung erhalten haben, unterliegt Ihre Verwendung der Software den darin enthaltenen Bedingungen. Es ist Ihnen nicht gestattet, die Pre-Release-Software unterzulizenzieren, per Leasing zur Verfügung zu stellen, zu verleihen, zu vermieten, weiterzugeben oder auf sonstige Weise Dritten zu überlassen. Sie verpflichten sich, bei Erhalt einer späteren unveröffentlichten Version der Pre-Release-Software bzw. bei Erhalt einer kommerziellen Vollversion von Altova (entweder als Einzelprodukt oder als Teil eines Gesamtprodukts) die zuvor von Altova erhaltene Pre-Release-Software zurückzugeben oder zu vernichten und den Bestimmungen des Lizenzvertrags für die betreffende spätere Version der Pre-Release-Software Folge zu leisten.

5. EINGESCHRÄNKTE GARANTIE UND HAFTUNGSBESCHRÄNKUNG

(a) **Eingeschränkte Garantie und Garantieansprüche der Kunden.** Altova garantiert der Person/Rechtspersönlichkeit, die ursprünglich die Lizenz für die Verwendung der Software gemäß den Bestimmungen dieses Lizenzvertrags erworben hat, dass (i) die Software über einen Zeitraum von neunzig (90) Tagen nach Erhalt im Wesentlichen in Übereinstimmung mit der dazugehörigen Dokumentation funktioniert, und (ii) die von Altova zur Verfügung gestellten Supportleistungen im Wesentlichen auf die in Abschnitt 6 dieses Vertrags niedergelegte Weise erfolgen. In einigen Ländern sind Beschränkungen über die Dauer einer stillschweigenden Garantie nicht zulässig, so dass die obigen Beschränkungen und Ausschlüsse eventuell für Sie nicht zutreffen. Im größtmöglichen rechtlich zulässigen Maß sind stillschweigende Garantien in Bezug auf die Software (sofern solche existieren) auf neunzig (90) Tage beschränkt. Die gesamte Haftung von Altova und dessen Lieferanten, sowie Ihre einzigen Garantieansprüche, sind nach dem Ermessen von Altova auf eine der beiden folgenden Optionen beschränkt: (i) Erstattung des Kaufpreises (wenn zutreffend), oder (ii) Reparatur oder Austausch der Software, die unter die eingeschränkte Garantie von Altova fällt und die unter Vorlage einer Kopie des Kaufbelegs bei Altova reklamiert wird. Diese eingeschränkte Garantie gilt nicht, wenn die Funktionalität der Software durch ein Versehen, durch Missbrauch, falsche Anwendung, Trojaner, Viren oder einen

sonstigen schädlichen externen Code beeinträchtigt wurde. Die Garantie für jede Ersatzsoftware erstreckt sich auf die Restdauer des ursprünglichen Garantiezeitraums oder auf dreißig (30) Tage, je nachdem, welcher Zeitraum länger ist. Die eingeschränkte Garantie gilt nicht für Test- und/oder Pre-Release-Software.

(b) **Keine weiteren Garantien, Haftungsausschluss.** MIT AUSNAHME DER VORSTEHEND ANGEFÜHRTEN EINGESCHRÄNKTEN GARANTIE UND DEN DIESBEZÜGLICHEN GARANTIEANSPRÜCHEN BESTEHEN SEITENS ALTOVA ODER DESSEN LIEFERANTEN KEINERLEI WEITEREN GARANTIEVERPFLICHTUNGEN. ALTOVA UND DESSEN LIEFERANTEN ÜBERNEHMEN KEINE GEWÄHR FÜR DIE ANWENDUNG ODER DIE ERGEBNISSE AUS DER NUTZUNG DER SOFTWARE. MIT AUSNAHME DER VORSTEHEND ANGEFÜHRTEN EINGESCHRÄNKTEN GARANTIE SOWIE IM HINBLICK AUF ALLE ANDEREN GEWÄHRLEISTUNGEN, BEDINGUNGEN, ZUSICHERUNGEN ODER ANSPRÜCHE, DIE NACH DER FÜR IHR LAND GELTENDEN RECHTSORDNUNG NICHT AUSGESCHLOSSEN ODER EINGESCHRÄNKT WERDEN KÖNNEN, SCHLIESSEN ALTOVA UND DESSEN LIEFERANTEN ALLE ANDEREN AUSDRÜCKLICHEN ODER STILLSCHWEIGENDEN GEWÄHRLEISTUNGEN, BEDINGUNGEN, ZUSICHERUNGEN UND ANSPRÜCHE AUS, DIE SICH AUS DEM GESETZ, DER RECHTSPRAXIS, EINEM GEWOHNHEITSRECHT, EINEM HANDELSBRAUCH ODER AUS SONSTIGEN GRÜNDEN ERGEBEN. ALTOVA UND DESSEN LIEFERANTEN SCHLIESSEN IM GRÖSSTMÖGLICHEN RECHTLICH ZULÄSSIGEN UMFANG ALLE ANDEREN AUSDRÜCKLICHEN UND STILLSCHWEIGENDEN GARANTIEN UND GEWÄHRLEISTUNGEN AUS. DIES BEINHALTET UNTER ANDEREM GARANTIEN IM HINBLICK AUF MARKTGÄNGIGKEIT, EIGNUNG FÜR EINEN BESTIMMTEN ZWECK, ZUFRIEDENSTELLENDE QUALITÄT, INFORMATIONSGEHALT ODER –RICHTIGKEIT, UNGESTÖRTE NUTZUNG, EIGENTUMSRECHT UND NICHTVERLETZUNG VON RECHTEN DRITTER UND DIE ERBRINGUNG ODER NICHTERBRINGUNG VON SUPPORTLEISTUNGEN IN BEZUG AUF DIE SOFTWARE. MIT DIESER EINGESCHRÄNKTEN GARANTIE WERDEN IHNEN BESTIMMTE RECHTE EINGERÄUMT. UNTER UMSTÄNDEN BESITZEN SIE NOCH ANDERE RECHTE, DIE JE NACH LAND/RECHTSORDNUNG UNTERSCHIEDLICH SEIN KÖNNEN.

(c) **Haftungsbeschränkung.** SOWEIT DIE ANWENDBAREN GESETZE DIES ZULASSEN, HAFTEN ALTOVA ODER DESSEN LIEFERANTEN, AUCH WENN EINE IM RAHMEN DER GARANTIE DURCHGEFÜHRTE ABHILFEMASSNAHME IHREN WESENTLICHEN ZWECK NICHT ERFÜLLT, IN KEINEM FALL FÜR KONKRETE, ZUFÄLLIG ENTSTANDENE, MITTELBARE, UNMITTELBARE ODER FOLGESCHÄDEN JEGLICHER ART (INSBESONDERE SCHÄDEN AUS ENTGANGENEM GEWINN, GESCHÄFTSUNTERBRECHUNGEN, VERLUST VON GESCHÄFTSINFORMATIONEN ODER ANDEREN FINANZIELLEN VERLUSTEN), DIE DURCH DIE NUTZUNG ODER DIE UNMÖGLICHKEIT DER NUTZUNG DER SOFTWARE ODER DIE NICHTERBRINGUNG VON SUPPORTLEISTUNGEN ENTSTANDEN SIND, SELBST WENN ALTOVA AUF DIE MÖGLICHKEIT EINES SOLCHEN SCHADENS HINGEWIESEN WURDE. IN JEDEM FALL IST DIE GESAMTHAFTUNG VON ALTOVA GEMÄSS DIESEM LIZENZVERTRAG AUF DIE HÖHE DES FÜR DAS SOFTWAREPRODUKT ENTRICHTETEN BETRAGES BESCHRÄNKT. Da es in einigen Ländern und Rechtsordnungen nicht zulässig ist, die Haftung für Schäden auszuschließen oder zu

 © 2007 Altova GmbH

beschränken, gilt die obige Einschränkung unter Umständen nicht für Sie. In solchen Ländern und Rechtsordnungen gilt die Beschränkung der Haftung durch Altova im größtmöglichen rechtlich zulässigen Umfang, und der Ausschluss bzw. die Beschränkung der in diesem Lizenzvertrag beinhalteten Garantien und Haftungen hat keine Auswirkung auf die Konsumentenschutzrechte von Personen, die Waren auf andere Weise als im Zuge ihrer geschäftlichen Tätigkeit erwerben. Der vorstehende Haftungssauschluss/die vorstehende Haftungsbeschränkung sind wesentliche Bestandteile des zwischen Ihnen und Altova abgeschlossenen Lizenzvertrags.

(d) **Ansprüche in Zusammenhang mit Urheberrechtsverletzungen.** Altova wird Sie gegenüber allen Forderungen, Prozessen oder Verfahren schad- und klaglos halten bzw. alle Ansprüche, Prozesse oder Verfahren beilegen, die Dritte mit dem Argument gegen Sie erheben, dass der Inhalt der Software gegen ein Urheberrecht verstößt oder ein geistiges oder sonstiges Eigentumsrecht verletzt, das durch das Recht der Vereinigten Staaten oder der Europäischen Union geschützt ist (insgesamt als „Ansprüche“ bezeichnet). Dies erfolgt – soweit nicht ausdrücklich etwas Anderes festgelegt ist – jedoch nur insoweit, als der betreffende Anspruch sich direkt aus der Verwendung der Software ergibt und nach Maßgabe der in Abschnitt 5 dieses Vertrags festgelegten Beschränkungen. Altova ist von jedem Anspruch innerhalb von zehn (10) Geschäftstagen, nachdem Sie erstmals davon benachrichtigt wurden, in Kenntnis zu setzen. Außerdem haben Sie mit Altova in angemessenem Umfang bei der Abwehr solcher Ansprüche zu kooperieren und Altova dabei zu unterstützen, ohne dass Sie dafür Kosten geltend machen können. Das Recht auf Entscheidungen in Bezug auf solche Ansprüche liegt allein bei Altova (dies beinhaltet auch, ohne darauf beschränkt zu sein, die Auswahl der Rechtsberater und das Recht, für Sie einen Vergleich zu den von Altova für zweckmäßig erachteten Bedingungen einzugehen). Sie können auf eigene Kosten einen Rechtsberater hinzuziehen und an den Verfahrens- oder Vergleichsverhandlungen teilnehmen. Altova kommt bis zu einer Höhe von insgesamt maximal dem Kaufpreis der Software für die Schäden, Kosten und Anwaltsgebühren auf, zu deren Bezahlung Sie in Zusammenhang mit solchen Ansprüchen verpflichtet werden (oder die Sie aufgrund eines Vergleichs zu entrichten haben), soweit diese nicht von einer Versicherung oder dritten Partei übernommen werden. Ist oder wird die Software Gegenstand von aufgrund einer Urheberrechtsverletzung vorgebrachten Ansprüchen, oder wird ihre Verwendung untersagt, oder ist es nach Ansicht des Rechtsberaters von Altova wahrscheinlich, dass ein solcher Umstand eintritt, so wird Altova versuchen, eine Beilegung herbeizuführen, indem alle wirtschaftlich vertretbaren Anstrengungen unternommen werden, um die Software zu modifizieren oder eine Lizenz für die weitere Verwendung der Software zu erwerben. Wenn es nach Ansicht des Rechtsberaters von Altova nicht möglich ist, den bevorstehenden oder bereits vorgebrachten Anspruch bzw. die Verfügung, mit der die Verwendung der Software untersagt wurde, durch angemessene Abänderung oder den Erwerb einer Lizenz beizulegen, so kann Altova diesen Lizenzvertrag ohne negative Konsequenzen für Altova beenden und Ihnen anteilig alle bereits an Altova entrichteten Gebühren zurückerstatten. MIT AUSNAHME DER VORSTEHEND ANGEFÜHRTEN BESTIMMUNGEN OBLIEGEN ALTOVA KEINERLEI HAFTUNGSVERPFLICHTUNGEN FÜR ANSPRÜCHE IN ZUSAMMENHANG MIT URHEBERRECHTSVERLETZUNGEN. Diese Haftungsverpflichtung gilt nicht für Urheberrechtsverletzungen, die ausschließlich auf vom Kunden eingebrachte Elemente zurückzuführen sind.

6. SUPPORT UND WARTUNG

Zusätzlich zu der Version des Softwareprodukts, für das Sie eine Lizenz erhalten haben, bietet

Altova eine Reihe von optionalen „Support- & Wartungspaketen“ („*Software & Maintenance Package“*, „SMP“) an, die Sie zusätzlich zu Ihrer Lizenz für die Software erwerben können. Der für das jeweilige SMP geltende Supportzeitraum (wie nachfolgend definiert) wird beim Kauf des SMP festgelegt. Die Support- und Wartungsservices und die Upgrades, die Ihnen zur Verfügung stehen, hängen davon ab, ob Sie ein SMP erwerben bzw. für welche Version des SMP Sie sich entscheiden.

(a) Wenn Sie kein SMP erwerben, erhalten Sie beim Kauf lediglich die Software, danach jedoch keinerlei Wartungsreleases oder Updates. Obwohl die Möglichkeit besteht, dass Altova Ihnen in Einzelfällen Wartungsreleases als kostenlose Zusatzleistung zur Verfügung stellt, sind in diesen Releases keine neuen Produktfeatures, die über das beim Kauf der Software bestehende Maß hinausgehen, beinhaltet. In jedem Fall erhalten Sie dreißig (30) Tage lang ab Kauf der Software (der „Supportzeitraum” im Sinne dieses Absatzes (a)) kostenlosen technischen Support von Altova. Außerdem kann Altova in Einzelfällen auch während des dreißig (30) Tage dauernden Evaluierungszeitraums technischen Support als kostenlose Zusatzleistung zur Verfügung stellen. Technischen Support erhalten Sie ausschließlich über ein webbasiertes Supportformular, wobei es keine garantierte Reaktionszeit gibt.

(b) Mit dem Erwerb eines SMP haben Sie während des dafür geltenden Supportzeitraums **Anspruch auf die von Ihnen erworbene Produktversion** sowie auf alle Wartungsreleases und Updates für diese Produktversion, die während des für Sie geltenden Supportzeitraumsfreigegeben werden. Während des Supportzeitraums Ihres SMP erhalten Sie auch Upgrades auf die entsprechende Produktversion der Software, mit der auf eine höhere Version der Software, für die Sie die Lizenz erworben haben, gewechselt wird und die während Ihres Supportzeitraums freigegeben werden. Die einzelnen Upgradeeditionen, auf die Sie innerhalb Ihres Supportzeitraums Anspruch haben, sind in dem von Ihnen erworbenen SMP im Detail angeführt. Software, die als gesondertes Produkt eingeführt wird, ist nicht im SMP enthalten. Wartungsreleases, Updates und Upgrades können neue Features enthalten, dies muss aber nicht der Fall sein. Darüber hinaus erhalten Sie während des Supportzeitraums bevorzugten technischen Support von Altova, und zwar ausschließlich über ein webbasiertes Supportformular. Altova wird alle wirtschaftlich vertretbaren Anstrengungen unternehmen, um alle Anfragen per E-Mail innerhalb von achtundvierzig (48) Stunden während der Geschäftszeiten (Montag bis Freitag, 8.00 bis 22.00 Uhr UTC, ausgenommen Feiertage in Österreich und den USA) zu beantworten. Außerdem wird Altova sich in angemessenem Umfang darum bemühen, Workarounds für Fehler, die bei der Software aufgetreten sind, zur Verfügung zu stellen.

Während des Supportzeitraums können Sie Altova Fehler oder Defekte in der Software melden. Wenn nach Altovas Einschätzung ein reproduzierbarer schwerwiegender Fehler vorliegt, der die Verwendung und Funktionalität der Software erheblich beeinträchtigt, wird Altova wirtschaftlich vertretbaren Anstrengungen unternehmen um Korrekturen oder provisorische Lösungen anzubieten, die in zukünftigen Updates oder Wartungsreleases enthalten sind. Diese Updates oder Wartungsreleases werden von Altova von Zeit zu Zeit zur Verfügung gestellt.

Es steht im Ermessen von Altova, einen schriftlichen Nachweis über von Ihnen festgestellte Fehler oder Funktionsstörungen zu verlangen oder Beispieldateien anzufordern, aus denen das aufgetretene Softwareproblem hervorgeht. In einem solchen Fall müssen Sie die angeforderten Nachweise oder Dateien, aus denen ausreichend detailliert hervorgeht, in welchen Aspekten Fehler bei der Software auftreten, so schnell wie möglich per E-Mail, Fax oder Expresspost mit Zustellung am nächsten Tag an Altova übermitteln. Bei der Diagnose oder Analyse von Fehlern haben Sie in zumutbarem Rahmen mit

 © 2007 Altova GmbH

Altova zu kooperieren. Fehlerbehebungen können in Wartungsreleases, Updates oder neuen Hauptversionen der Software enthalten sein. Altova ist nicht verpflichtet, unwesentliche Fehler, d.h. Fehler, die die Benutzung der Software nicht wesentlich beeinträchtigen, zu beheben. Unabhängig davon, ob Sie das Support- und Wartungspaket erworben haben, wird technischer Support ausschließlich für Fragen oder Probleme angeboten, die sich unmittelbar aus oder in Zusammenhang mit der Software ergeben. Sie erhalten von Altova unter keinen Umständen Beratung, Unterstützung oder Hilfestellung, die allgemeiner Natur ist und nicht in unmittelbarem Zusammenhang mit der Software steht.

Bei der Aktualisierung der Software kann es unter Umständen vorkommen, dass vor der Installation auch Software aktualisiert werden muss, die nicht unter diesen Lizenzvertrag fällt. Für Updates von Betriebssystem- und Anwendungssoftware, die nicht ausdrücklich Gegenstand dieses Lizenzvertrags ist, sind ausschließlich Sie verantwortlich. Solche Updates sind nicht im Umfang dieser Lizenz enthalten und werden nicht von Altova zur Verfügung gestellt. Die Erfüllung der in diesem Abschnitt 6 festgelegten Verpflichtungen durch Altova versteht sich unter der Bedingung, dass Sie die Software ordnungsgemäß verwenden und diesen Lizenzvertrag ausnahmslos jederzeit einhalten. Altova ist nicht verpflichtet, technischen Support zu leisten, wenn nach Ansicht von Altova die Fehlfunktion der Software auf einen der folgenden Gründe zurückzuführen ist: (i) Fehler, die durch die Verlegung der Software auf einen anderen Rechner oder Speicherort hervorgerufen werden; (ii) Änderungen, Modifikationen oder Versuche, die Software abzuwandeln, die von Altova nicht schriftlich genehmigt wurden; (iii) äußere Einflüsse auf die Software, wie z.B. Naturkatastrophen, Stromausfälle, Stromschwankungen oder Computerausfälle; (iv) Ihr Versäumnis die Software auf dem von Altova festgelegten Release Level zu halten; oder (v) Nutzung der Software ohne vorherige Genehmigung durch Altova zusammen mit einer anderen Software. Sie alleine sind dafür verantwortlich: (i) allen Betriebs- und Fehlerbehebungsanleitungen von Altova Folge zu leisten, Altova unverzüglich von Fehlern oder Defekten an der Software zu informieren und Altova eine genaue Beschreibung dieser Fehler und/oder Defekte zu liefern; (ii) für den Schutz Ihrer vertraulichen Informationen zu sorgen; (iii) Datensicherungssysteme und -abläufe für die Wiederherstellung verlorener oder geänderter Dateien, Daten oder Programme einzurichten und anzuwenden.

7. AKTIVIERUNG DER SOFTWARE, UPDATES UND LIZENZÜBERWACHUNG

(a) **Lizenzüberwachung.** Altova verfügt über ein integriertes Lizenzüberwachungsmodul, das Ihnen hilft, unbeabsichtigte Verletzungen dieses Lizenzvertrags zu vermeiden. Dabei kann Altova zur Lizenzüberwachung von verschiedenen installierten Versionen der Software Ihr internes Netzwerk verwenden.

(b) **Aktivierung der Software. Die Software von Altova kann Ihr internes Netzwerk und Ihre Internetverbindung verwenden, um Angaben über Ihre Lizenz im Zuge der Installation, Registrierung, Benutzung oder Aktualisierung an einen Altova-Lizenzserver zu übertragen und diese zu verifizieren, um auf diese Weise eine nicht lizenzierte oder illegale Verwendung der Software zu verhindern und den Kundenservice von Altova weiter zu verbessern. Die Aktivierung erfolgt über einen Austausch von lizenzbezogenen Daten zwischen Ihrem Computer und dem Altova-Lizenzserver. Sie stimmen dieser Vorgangsweise von Altova zu und verpflichten sich, allen diesbezüglichen Vorgaben Folge zu leisten.**

(c) **LiveUpdate**. Altova stellt Ihnen ein neues LiveUpdate Benachrichtigungsservice zur Verfügung, welches kostenlos ist. Altova kann Ihr internes Netzwerk und Ihre Internetverbindung verwenden, um Angaben über Ihre Lizenz an einen LiveUpdate-Server von Altova zu übertragen, um Ihre

Lizenz in gewissen Zeitabständen zu verifizieren und festzustellen, ob ein Update für Sie verfügbar ist.

(d) **Verwendung von Daten.** Der gesamte Wortlaut der Datenschutzbestimmungen von Altova kann unter http://www.altova.com/privacy eingesehen werden. Die Datenschutzbestimmungen sind durch Bezugnahme Teil dieses Lizenzvertrags. Durch Ihre Zustimmung zu den Bestimmungen dieses Lizenzvertrags bzw. durch Benutzung der Software erklären Sie sich damit einverstanden, dass Altova für die in diesem Lizenzvertrag und/oder in den Datenschutzbestimmungen (in der jeweils geltenden Fassung) genannten Zwecke Daten erhebt, verarbeitet und weitergibt. Benutzer in Europa haben Kenntnis davon und erklären sich damit einverstanden, dass Informationen zu ihrer Person in den Vereinigten Staaten zu den in diesem Vertrag angeführten Zwecken verwendet werden. Altova behält sich das Recht vor, diese Bestimmung des Lizenzvertrags und/oder der Datenschutzbestimmungen jederzeit zu ändern. Wir legen Ihnen nahe, die auf der Website von Altova veröffentlichten Datenschutzbestimmungen von Zeit zu Zeit erneut durchzulesen.

8. DAUER UND BEENDIGUNG

Dieser Lizenzvertrag wird folgendermaßen beendet: (a) indem Sie Altova eine schriftliche Benachrichtigung von der Vertragsbeendigung übermitteln; oder (b) durch Altova in Form einer diesbezüglichen schriftlichen Benachrichtigung, wenn Sie gegen diesen Lizenzvertrag verstoßen und diesen Verstoß nicht innerhalb von 10 (zehn) Tagen, nachdem Altova Ihnen dies mitgeteilt hat, berichtigen; oder (c) auf Verlangen eines autorisierten Vertriebshändlers von Altova, wenn Sie eine Lizenzzahlung nicht entrichten oder sonstige Zahlungsverpflichtungen nicht einhalten. Der Lizenzvertrag einer früheren Version der Software, für die Sie ein Upgrade oder ein Update durchgeführt haben, wird mit Ihrer Zustimmung zu dem Lizenzvertrag, der für das Upgrade oder Update gilt, automatisch beendet. Bei Beendigung des Lizenzvertrags dürfen Sie die gesamte Software, die Gegenstand der Lizenz ist, nicht mehr verwenden, müssen alle Kopien, die in Ihrem Besitz oder Einflussbereich stehen, vernichten, und müssen in zumutbarem Rahmen alle von Altova geforderten Maßnahmen ergreifen, um sicherzustellen, dass keine Kopien der Software in Ihrem Besitz oder Einflussbereich verbleiben. Die in den Abschnitten 1(g), (h), (i), 2, 5(b), (c), 9, 10 und 11 niedergelegten Bestimmungen bleiben auch nach Beendigung dieses Lizenzvertrags weiterhin aufrecht.

9. EINGESCHRÄNKTE RECHTE UND EXPORTBESCHRÄNKUNGEN

Die Entwicklung der Software wurde ausschließlich privat finanziert. Bei der Software handelt es sich um kommerzielle Computersoftware, die mit **eingeschränkten Rechten** ausgestattet ist. Die Verwendung, Vervielfältigung oder Weitergabe der Software durch die US-Regierung, eine Behörde oder einen Kooperationspartner der US-Regierung unterliegt den Beschränkungen im Rahmen dieses Vertrags sowie den Beschränkungen nach FAR 12.211 und 12.212 (48 C.F.R. §12.211 und 12.212) bzw. DFARS 227.7202 (48 C.F.R. §227-7202). Dabei wird eine Lizenz für kommerzielle Computersoftware und kommerzielle Computerdokumentation an die US-Regierung als Endnutzer ausschließlich in Form einer Lizenz für kommerziell genutzte Güter erteilt, weshalb es sich bei den damit verbundenen Rechten um dieselben Rechte handelt, die allen anderen Endnutzern im Rahmen dieses Lizenzvertrags gewährt werden. Beim Hersteller handelt es sich um Altova GmbH, Rudolfsplatz, 13a/9, A-1010 Wien, Österreich/EU. Die Software und/oder Dokumentation darf ausschließlich unter Einhaltung aller anwendbaren Exportbestimmungen der Vereinigten Staaten von Amerika sowie des Landes, in dem Sie die Software erhalten haben, verwendet, exportiert oder wiedereingeführt werden. Insbesondere darf die

Software und/oder Dokumentation nicht (i) in ein Land exportiert oder wiedereingeführt werden, über das die Vereinigten Staaten ein Embargo verhängt haben, oder einem Staatsangehörigen oder Bewohner eines solchen Landes überlassen werden; oder (ii) einer Person überlassen werden, die auf der Liste der Specially Designated Nationals des U.S. Treasury Department oder dem Table of Denial Orders des U.S. Department of Commerce verzeichnet sind. Indem Sie die Software benutzen, erklären Sie, dass Sie weder in einem dieser Länder ansässig sind noch seiner Kontrolle unterliegen, noch ein Staatsangehöriger oder Bewohner eines dieser Länder sind, noch auf einer der vorstehend erwähnten Listen genannt werden.

10. SOFTWARE DRITTER

Die dieser Lizenz unterliegende Software kann Software Dritter enthalten, für die ergänzende Vermerke und/oder Nutzungsbedingungen gelten. Diese Vermerke und ergänzenden Nutzungsbedingungen für die Software Dritter können über unsere Website unter http://www.altova.com/legal_3rdparty.html eingesehen werden und sind durch Bezugnahme Teil dieses Lizenzvertrags. Indem Sie Ihre Zustimmung zu den Bedingungen dieses Vertrags erteilen, stimmen Sie auch automatisch allen ergänzenden Bestimmungen, die möglicherweise darin enthalten sind, zu.

11. ALLGEMEINE BESTIMMUNGEN

Wenn Sie sich in der Europäischen Union befinden und die Software in der Europäischen Union und nicht in den Vereinigten Staaten verwenden, unterliegt dieser Lizenzvertrag dem Recht der Republik Österreich (unter Ausschluss von dessen Verweisungsnormen und der UN-Kaufrechtskonvention). Sie erklären sich ausdrücklich damit einverstanden, dass alle Streitigkeiten oder Konflikte mit Altova, die in Zusammenhang mit Ihrer Verwendung der Software stehen, in die alleinige Zuständigkeit des Handelsgerichts Wien fallen. Sie erklären sich weiters ausdrücklich damit einverstanden, dass Sie bezüglich solcher Streitigkeiten oder Konflikte der persönlichen Zuständigkeit des Handelsgerichts Wien unterstellt sind.

Wenn Sie sich in den Vereinigten Staaten befinden und die Software in den Vereinigten Staaten verwenden, unterliegt dieser Lizenzvertrag dem Recht des Commonwealth of Massachusetts, USA (unter Ausschluss von dessen Verweisungsnormen und der UN-Kaufrechtskonvention). Sie erklären sich ausdrücklich damit einverstanden, dass alle Streitigkeiten oder Konflikte mit Altova, die in Zusammenhang mit Ihrer Verwendung der Software stehen, in die alleinige Zuständigkeit der einzel- und bundesstaatlichen Gerichte im Bundesstaat Massachusetts fallen. Sie erklären sich weiters ausdrücklich damit einverstanden, dass Sie bezüglich solcher Streitigkeiten oder Konflikte der persönlichen Zuständigkeit der einzel- und bundesstaatlichen Gerichte im Bundesstaat Massachusetts unterstellt sind.

Wenn Sie sich nicht in der Europäischen Union oder den Vereinigten Staaten befinden und die Software nicht in den Vereinigten Staaten verwenden, unterliegt dieser Lizenzvertrag dem Recht der Republik Österreich (unter Ausschluss von dessen Verweisungsnormen und der UN-Kaufrechtskonvention). Sie erklären sich ausdrücklich damit einverstanden, dass alle Streitigkeiten oder Konflikte mit Altova, die in Zusammenhang mit Ihrer Verwendung der Software stehen, in die alleinige Zuständigkeit des Handelsgerichts Wien fallen. Sie erklären sich weiters ausdrücklich damit einverstanden, dass Sie bezüglich solcher Streitigkeiten oder Konflikte der persönlichen Zuständigkeit des Handelsgerichts Wien unterstellt sind. Die Anwendung von Verweisungsnormen einer Rechtsordnung sowie des UN-Abkommens zum internationalen Warenkauf (CISG) auf diesen Lizenzvertrag wird

ausdrücklich ausgeschlossen.

Dieser Lizenzvertrag enthält die gesamte Vereinbarung zwischen den Parteien in Bezug auf den Vertragsgegenstand und tritt an die Stelle aller diesbezüglichen früheren mündlichen oder schriftlichen Vereinbarungen zwischen den Parteien. Benachrichtigungen oder sonstige Mitteilungen im Rahmen dieses Lizenzvertrags müssen schriftlich erfolgen und gelten als ordnungsgemäß übermittelt, wenn sie per Einschreiben mit Rückschein oder per Kurierdienst mit Zustellung am nächsten Tag an die auf der Website von Altova angegebene Adresse (wenn Altova der Empfänger ist) bzw. an die in den Aufzeichnungen von Altova für Sie eingetragene Adresse (wenn Sie der Empfänger sind) oder aber an eine andere, von den Vertragspartnern festgelegte und auf die vorstehend beschriebene Weise bekannt gegebene Adresse geschickt werden. Dieser Lizenzvertrag ist für die Vertragspartner verbindlich und geht auf ihre Erben, persönlichen und rechtlichen Vertreter, verbundenen Unternehmen, Rechtsnachfolger und zulässigen Abtretungsempfänger über. Die Nichtdurchsetzung oder Nichtausübung von Rechten oder Rechtsmitteln unter diesem Lizenzvertrag durch die Vertragspartner stellt keinen Verzicht auf ein solches Recht oder Rechtsmittel dar und beeinträchtigt in keiner Weise das Recht des betreffenden Vertragspartners, ein solches Recht oder Rechtsmittel sowie alle anderen Rechte oder Rechtsmittel aus diesem Lizenzvertrag später durchzusetzen bzw. auszuüben. Eine Änderung dieses Lizenzvertrags ist nur Form einer schriftlich niedergelegten Vereinbarung möglich, die von beiden Vertragspartnern unterzeichnet wird. Wenn eine Zuwiderhandlung gegen die Bestimmungen dieses Lizenzvertrags durch einen der Vertragspartner erfolgt ist oder droht, so kann der jeweils andere Vertragspartner alle ihm zustehenden Rechte und Rechtsmittel geltend machen. Jeder Vertragspartner ist ordnungsgemäß befugt und ermächtigt, in diesen Lizenzvertrag einzutreten und die daraus erwachsenden Verpflichtungen zu erfüllen. Sollte eine Bestimmung dieses Lizenzvertrags für unwirksam, rechtswidrig oder undurchführbar erklärt werden, so wird dadurch die Wirksamkeit der übrigen Bestimmungen nicht berührt, und dieser Lizenzvertrag bleibt im größtmöglichen rechtlich zulässigen Umfang wirksam und gültig. Die Vertragspartner haben die vorstehenden Vertragsbedingungen zur Kenntnis genommen und erklären sich ausdrücklich damit einverstanden.

Letzte Aktualisierung: 2006-09-05

© 2007 Altova GmbH

Index

.

.NET,
Integration von XMLSPY mit, 226
und SOAP Debugger, 234
und XSLT Debugger, 231
Unterschiede zur XMLSPY Standalone-Version, 227

2

2005R3,
compatibility mode, 336

A

Abfrage,
Oracle XML DB, 516
siehe Datenbank abfragen, 482
siehe Datenbankabfrage (Fenster(, 482
siehe XQuery, 482
Abfragefenster,
im Fenster Datenbankabfrage, 489
Abfragen,
für DB-Anzeige in der Authentic-Ansicht, 210
Ableiten eines Typs, 164
ActiveX controls,
support, 648
ADO,
Konvertierung von Datentypen in XML-Schema Generierung von DB, 1008
ADO-Verbindungen, 476
Aktiviere Versionskontrolle, 387
Aktuelle Execution Nodes einblenden,
XSLT Debugger, 462
Aktuellste Version holen, 388
Alle schließen, 362
Alles markieren,
Befehl, 373
Altova,
Produktfamilie, 8
Altova XML Parser,
Info, 1021
Altova XSLT 1.0-Prozessor,
Einschränkungen und implementierungsspezifisches Verhalten, 990
Altova XSLT 2.0-Prozessor,
allgemeine Informationen, 992
Informationen, 992
Altova-Prozessoren,
in Altova-Produkten, 1022
Anbieter,
Versionskontrolle, 383
Änderungen verfolgen, 587
Angedocktes Fenster, 602
Anhängen, 408
Zeile (Dokument Editor), 470
Anpassen,
Kontextmenü, 581
Kontextmenü 'Anpassen', 585
Makros, 583
Menü, 581
Symbolleiste/Menü Befehle, 574
Ansicht,
ausklappen, 543
Browser-Ansicht, 543
einklappen, 544
Enhanced Grid-Ansicht, 542
erweitern, 544
Gehe zu Datei, 545
Gehe zu Zeile/Zeichen, 545
Menü, 542
Optimale Breite, 544
reduzieren, 544
Schema Design-Ansicht, 543
Textansicht, 542
XMLSpy in Eclipse, 250
Ansicht konfigurieren,
Dialogfeld in der Content Model-Ansicht, 441
Ansicht wechseln,
zur Authentic-Ansicht, 191
Anwendung beenden, 368
Anzeige-Einstellungen, 596
Anzeigen, 602, 603
API,
accessing, 967
documentation, 662
JAVA, 907

© 2007 Altova GmbH

API,
JAVA Classpath, 907
overview, 663
Application,
ActiveDocument, 687
AddMacroMenuItem, 688
Application, 688
ClearMacroMenu, 688
CurrentProject, 688
Dialogs, 689
Documents, 689
GetDatabaseImportElementList, 689
GetDatabaseSettings, 690
GetDatabaseTables, 690
GetExportSettings, 691
GetTextImportElementList, 691
GetTextImportExportSettings, 692
ImportFromDatabase, 692
ImportFromSchema, 693
ImportFromText, 694
ImportFromWord, 695
NewProject, 695
OnBeforeOpenDocument, 685
OnBeforeOpenProject, 686
OnDocumentOpened, 686
OnProjectOpened, 687
OpenProject, 695
Parent, 696
Quit, 696
ReloadSettings, 696
RunMacro, 697
ScriptingEnvironment, 697
ShowApplication, 697
ShowForm, 698
URLDelete, 698
URLMakeDirectory, 699
WarningNumber, 699
WarningText, 699
Application name, 336
Application-level,
integration of XMLSpy, 942
Arabisch, 1024
Arbeitsspeicher-Anforderungen, 1020
Arbeitsverzeichnis,
Versionskontrolle, 385
Archivansicht, 221
Befehlsschaltflächen, 222
Benutzeroberfläche, 222
Operationen, 224
Arial Unicode MS, 1023
Assigning,
scripting projects, 614
ATL,
plug-in sample files, 652
Atomisierung von Nodes,
in XPath 2.0- und XQuery 1.0-Auswertung, 998
ATTLIST Deklaration,
als Child in Grid-Ansciht hinzufügen, 415
in Grid-Ansicht anhängen, 412
in Grid-Ansicht einfügen, 408
Konvertieren in (Grid-Ansicht), 417
Attribut,
in Grid-Ansicht anhängen, 409
Konvertieren in (Grid-Ansicht), 416
Attribute,
als Child in Grid-Ansicht hinzufügen, 413
in der Content Model-Ansicht ein- und ausblenden, 47
in der Schema-Übersicht für eine Komponente definieren, 150
in Grid-Ansicht einfügen, 405
in Schemadefinitionen, 47
AttributeFormDefault,
Einstellungen in der Schema Design-Ansicht, 438
Attribut-Eingabehilfe,
in der Authentic-Ansicht, 195
Attribut-Vorschau, 592
Attributwerte,
in der Authentic-Ansicht eingeben, 93
Ausblenden,
Markup, 470
Auschecken, 388
Auschecken rückgängig, 390
Ausgabefenster, 19, 109
Anzeige ein-/aus, 602
Ausgabeformatierung, 587
Ausgliedern,
aus Versionskontrolle, 392
Ausschneiden,
Befehl, 369
Aussteigen,
XSLT Debugger, 462
Authentic,
CDATA-Abschnitte, 90
Daten eingeben, 90
Dateneingabe-Steuerelemente, 90
DB-Daten bearbeiten, 467

© 2007 Altova GmbH

Authentic,
Eingabehilfen, 85
Elemente anwenden, 87
Elemente löschen, 87
Entities, 90
Kontextmenü, 85
Markup-Tags anzeigen, 85
Menü, 466
Nodes einfügen, 87
Nodes entfernen, 87
Nodes hinzufügen, 87
Projektfenster, 190
Sonderzeichen, 90
Tabellen, 87
Authentic-Ansicht,
Anzeige der Markup-Symbole, 191, 193
Anzeige des Dokuments, 193
Attribut-Eingabehilfe, 195
Attributwerte eingeben, 93
Benutzeroberfläche, 188
dynamische Tabellen bearbeiten, 191
Eingabehilfen, 190, 195
Element-Eingabehilfe, 195
Entities einfügen, 93
Entities-Eingabehilfe, 195
Hauptfenster, 190, 193
Kontextmenü, 195
Kontextmenüs, 195, 199
Menüleiste, 190
neue XML-Datei öffnen, 466
SPS-Tabellen, 202
Statusleiste, 190
Symbolleiste, 190
Symbolleistenschaltflächen, 191
Tabellen (SPS und XML), 202
Text formatieren, 191
Tutorial, 83
Übersicht über die GUI, 190
Verwendung von XML-Tabellen, 203
Verwendung wichtiger Features, 202
Voraussetzungen für das Tutorial, 83
Vorlage, 84
wechseln zur, 191, 543
XML-Dokument drucken, 94
XML-Dokument öffnen, 84
XML-Tabellen, 203
XML-Tabellensymbole, 207
XPath zum ausgewählten Node, 190
AuthenticDataTransfer,
dropEffect, 700
getData, 701
ownDrag, 701
type, 701
Authentic-Menü,
Anzeige von Markup-Symbolen, 191
dynamische Tabellen bearbeiten, 191
AuthenticRange,
AppendRow, 703
Application, 703
CanPerformAction, 704
CanPerformActionWith, 704
Close, 705
CollapsToBegin, 705
CollapsToEnd, 705
Copy, 705
Cut, 706
Delete, 706
DeleteRow, 706
DuplicateRow, 707
ExpandTo, 707
FirstTextPosition, 707
FirstXMLData, 708
FirstXMLDataOffset, 709
GetElementAttributeNames, 710
GetElementAttributeValue, 710
GetElementHierarchy, 711
GetEntityNames, 711
Goto, 711
GotoNext, 712
GotoNextCursorPosition, 712
GotoPrevious, 713
GotoPreviousCursorPosition, 713
HasElementAttribute, 714
InsertEntity, 714
InsertRow, 715
IsCopyEnabled, 715
IsCutEnabled, 715
IsDeleteEnabled, 716
IsEmpty, 716
IsEqual, 716
IsFirstRow, 716
IsInDynamicTable, 717
IsLastRow, 717
IsPasteEnabled, 717
IsTextStateApplied, 717
LastTextPosition, 718

© 2007 Altova GmbH

AuthenticRange,
LastXMLData, 718
LastXMLDataOffset, 719
MoveBegin, 720
MoveEnd, 720
MoveRowDown, 721
MoveRowUp, 721
Parent, 721
Paste, 722
PerformAction, 722
Select, 723
SelectNext, 723
SelectPrevious, 724
SetElementAttributeValue, 724
SetFromRange, 725
Text, 726
AuthenticView, 739
Application, 735
AsXMLString, 735
DocumentBegin, 735
DocumentEnd, 735
Event, 736
Goto, 736
IsRedoEnabled, 737
IsUndoEnabled, 737
MarkupVisibility, 737
OnBeforeCopy, 727
OnBeforeCut, 728
OnBeforeDelete, 728
OnBeforeDrop, 728
OnBeforePaste, 730
OnDragOver, 730
OnKeyboardEvent, 731
OnMouseEvent, 732
OnSelectionChanged, 734
Parent, 737
Print, 738
Redo, 738
Selection, 738
Undo, 739
WholeDocument, 739
XMLDataRoot, 740
Autokomplettierung,
in der Textansicht, 117
Auto-Komplettierung,
Text-Ansicht Ein/Aus, 591
Autokomplettierung in SQL-Scripts, 489
Automatische Validierung, 589
Automatische Validierung ausschalten, 589

B

Background,
gradient, 336
Base package, 336
Bearbeiten,
Makro Schaltfläche, 585
Menü, 369
Bearbeiten rückgängig, 390
Bearbeitet mit XMLSPY, 587
Beenden,
Debuggersitzung, 461
Befehl,
hinzufügen zu Symbolleiste/Menü, 574
Kontextmenü, 581
löschen aus Menü, 581
Menü zurücksetzen, 581
Befehle,
Auflistung in Tastaturzuordnung, 605
Beispieldateien,
Tutorial, 21
Benutzerdefiniert,
Wörterbuch, 559
Benutzerhandbuch, 3, 6
Benutzerhandbücher,
für Altova-Produkte, 8
Benutzeroberfläche,
Übersicht, 14
Bestellen der Altova-Software, 606
Betriebssysteme für Altova-Produkte, 1020
Bibliotheksmodule,
in XQuery-Dokument, 995
Big-endian, 598
Breakpoint,
Dialogfeld, 463
Breakpoint Aus - XSLT Debugger, 463
Breakpoint Ein - XSLT Debugger, 463
Breakpoints,
im XSLT/XQuery Debugger, 271
Breakpoints dialog, 256
Browser, 592
Ansicht, 543
Menü, 547
Browser-Ansicht,

© 2007 Altova GmbH

Browser-Ansicht,
Beschreibung, 220
Inhalt aktualisieren, 547
Laden einer Seite abbrechen, 547
neues Fenster, 547
Schriftgröße, 547
weiter, 547
zurück, 547
Browser-Fenster,
im Fenster "Datenbankabfrage", 485

C

C#,
code, 303
integration of XMLSpy, 946, 947, 948, 949
settings, 336
C++,
code, 303
settings, 336
Call Stack-Fenster,
in XSLT/XQuery Debugger, 266
Callgraph-Profilerstellung, 279
CDATA,
als Child in Grid-Ansicht hinzufügen, 413
in Grid-Ansicht anhängen, 410
in Grid-Ansicht einfügen, 406
Konvertieren in (Grid-Ansicht), 416
Child anhängen,
in Grid-Ansicht, 412
Class,
JAVA, 907
Class ID,
in XMLSpy integration, 942
ClassPath statement, 907
Code,
built in types, 347
SPL, 337
Code compatibility,
v2005R3, 336
Code Generator, 303, 437
CodeGeneratorDlg,
Application, 741
CPPSettings_DOMType, 741
CPPSettings_LibraryType, 742
CPPSettings_UseMFC, 742
CSharpSettings_ProjectType, 743
OutputPath, 743
OutputPathDialogAction, 743
OutputResultDialogAction, 743
Parent, 744
ProgrammingLanguage, 744
PropertySheetDialogAction, 744
TemplateFileName, 745
Codepage, 593
Collations,
in XPath 2.0, 998
in XQuery-Dokument, 995
COM-API,
documentation, 662
Comment,
als Child in Grid-Ansicht hinzufügen, 414
in Grid-Ansicht anhängen, 410
in Grid-Ansicht einfügen, 406
Konvertieren in (Grid-Ansicht), 416
Compatibility mode,
v2005R3, 336
Compiler,
settings, 336
ComplexType,
Definition erweitern, 38
ComplexTypes,
in Schema-Definitionen, 38
Configure,
XMLSPY UI, 649
Content Model,
Attribute ein- und ausblenden, 47
Diagramm speichern, 439
einfaches Content Model erstellen, 27
Content Model-Ansicht, 23
ausführliche Beschreibung, 154
Grundzüge, 145
konfigurieren, 441
Content Models,
von Schema-Komponenten, 154
Copyright-Informationen, 1026
count() function in XPath 2.0,
siehe fn:count(), 998
count()-Funktion,
in XPath 1.0, 990
CR&LF, 587
CSS,
Autokomplettierung, 131
Dokumentübersicht, 131

CSS,
Eigenschaften, 131
Syntaxfärbung, 131
CSV-Datei,
als XML importieren, 523
CustomCatalog, 422
CVS, 383

D

Darstellung, 592
DatabaseConnection,
ADOConnection, 745
AsAttributes, 746
CreateMissingTables, 746
CreateNew, 747
DatabaseKind, 747
ExcludeKeys, 747
File, 747
IncludeEmptyElements, 748
NumberDateTimeFormat, 748
ODBCConnection, 748
SQLSelect, 749
TextFieldLen, 749
Datei,
Alle schließen, 362
Alles speichern, 364
als Mail senden, 365
Druckoptionen, 366
in Eclipse importieren, 241
Kodierung, 362
Menü, 356
Neu laden, 361
Öffnen, 359
Optionen beim Öffnen, 587
Register, 587
schließen, 362
speichern, 363
Speichern unter, 363
Speichern unter URL, 363
Standard-Encoding, 598
URL öffnen, 359
Datei holen,
Versionskontrolle, 388
Dateiarten, 589
Dateien,
Vergleichsoptionen, 571
zuletzt verwendete, 368
zwei vergleichen, 566
Dateien speichern,
Kodierung, 362
Dateien vergleichen, 566
Optionen, 571
Dateierweiterungen,
für XQuery-Dateien, 120
Dateipfad,
einfügen, 372
Dateipfad einfügen, 372
Daten,
in und aus externen Applikationen importieren und exportieren, 140
Datenbank,
ADO-Verbindungen, 476
bestehende Verbindungen, 475
Datensätze bearbeiten, 492
DB auf Basis von Schema erzeugen, 534
ODBC-Verbindungen, 480
Oracle XML DB, 516
Struktur als XML-Schema importieren, 529
Verbindung herstellen, 472
Verbindungsassistent, 473
XML-Daten exportieren, 539
Datenbank abfragen (Befehl), 482
Datenbank/Tabellenansicht,
Verwendung, 70
Datenbank/Tabellen-Ansicht,
in der Enhanced Grid-Ansicht, 140
Datenbankabfrage,
Abfrage erstellen, 492
Browser-Fenster im Fenster Datenbankabfrage, 485
Ergebnisse, 492
Ergebnisse bearbeiten, 492
Meldungsregister, 492
Verbindung zur DB für Abfrage herstellen, 483
Datenbankabfrage (Fenster), 482
Datenbanken,
in Authentic-Ansicht bearbeiten, 467
siehe auch DB, 209
Datensätze bearbeiten, 492
Datentypen,
für DB, die von XML-Schemas generiert wurden, 1010
in XPath 2.0 und XQuery 1.0, 998
Konvertierung von DB in XML-Schema, 1002
siehe auch XML-Schema-Datentypen, 1003

© 2007 Altova GmbH

Datumswahl,
in der Authentic-Ansicht verwenden, 216
DB,
Abfragen erstellen, 210
Abfragen in der Authentic-Ansicht, 209
Anzeige in der Authentic-Ansicht filtern, 210
Bearbeiten in der Authentic-Ansicht, 209
Datentypen von XML-Schema, 1010
Datentyp-Konvertierung in XML-Schema, 1002
in der Authentic-Ansicht bearbeiten, 214
Navigieren in Tabellen in der Authentic-Ansicht, 209
Parameters in DB-Abfragen, 210
DB XML,
XML Schemas zuweisen für IBM DB2, 499
XML-Schemas verwalten, für IBM DB2, 495
db2-fn:sqlquery, 126
db2-fn:xmlcolumn, 126
DBUri,
Datenbank abfragen, 516
Deaktiviere Versionskontrolle, 387
Debugger,
aktuelle Execution Nodes einblenden, 462
aussteigen, 462
Breakpoint Ein/Aus, 463
Breakpoint einfügen/löschen, 462
Debug-Fenster, 464
Dialogfeld Breakpoints, 463
einsteigen, 461
Einstellungen, 464
enable/disable tracepoint, 463
insert/remove tracepoint, 462
Sitzung beenden, 461
SOAP Debugger, 556
SOAP in Eclipse, 247
SOAP Kommunikation - Beispiel, 294
überspringen, 462
XSLT Debugger erneut starten, 461
XSLT Debugger starten, 461
XSLT Debugger stoppen, 461
XSLTin Eclipse, 245
Debugger -Fenster,
Anordnung, 268
deep-equal() function in XPath 2.0,
siehe fn:deep-equal(), 998
Default Functions Namespace,
für XPath 2.0- und XQueyr 1.0-Ausdrücke, 998
in XSLT 2.0 Stylesheets, 992
Delete,
Application.URLDelete, 698
Details-Eingabehilfe, 27
Diagramm anzeigen, 445
Dialogs,
Application, 750
CodeGeneratorDlg, 750
DTDSchemaGeneratorDlg, 751
FileSelectionDlg, 750
GenerateSampleXMLDlg, 751
Parent, 750
SchemaDocumentationDlg, 751
directories,
creating with Application.URLMakeDirectory, 699
Disable,
tracepoint - XSLT debugger, 463
Diskussionsforen,
für XML-Probleme, 8
Dll,
compiler settings, 336
DocEditEvent (obsolete),
altKey (obsolete), 851
altLeft (obsolete), 852
button (obsolete), 853
cancelBubble (obsolete), 854
clientX (obsolete), 855
clientY (obsolete), 856
ctrlKey (obsolete), 857
ctrlLeft (obsolete), 858
dataTransfer (obsolete), 859
fromElement (obsolete), 860
keyCode (obsolete), 860
propertyName (obsolete), 861
repeat (obsolete), 861
returnValue (obsolete), 861
shiftKey (obsolete), 862
shiftLeft (obsolete), 863
srcElement (obsolete), 864
type (obsolete), 865
DocEditView (obsolete),
ApplyTextState (obsolete), 871
CurrentSelection (obsolete), 872
EditClear (obsolete), 872
EditCopy (obsolete), 873
EditCut (obsolete), 873
EditPaste (obsolete), 874
EditRedo (obsolete), 874
EditSelectAll (obsolete), 875
EditUndo (obsolete), 875

© 2007 Altova GmbH

DocEditView (obsolete),
event (obsolete), 876
GetAllowedElements (obsolete), 876
GetNextVisible (obsolete), 879
GetPreviousVisible (obsolete), 880
IsEditClearEnabled (obsolete), 881
IsEditCopyEnabled (obsolete), 881
IsEditCutEnabled (obsolete), 882
IsEditPasteEnabled (obsolete), 882
IsEditRedoEnabled (obsolete), 883
IsEditUndoEnabled (obsolete), 883
IsRowAppendEnabled (obsolete), 884
IsRowDeleteEnabled (obsolete), 884
IsRowDuplicateEnabled (obsolete), 885
IsRowInsertEnabled (obsolete), 885
IsRowMoveDOwnEnabled (obsolete), 886
IsRowMoveUpEnabled (obsolete), 886
IsTextStateApplied (obsolete), 887
IsTextStateEnabled (obsolete), 887
LoadXML (obsolete), 888
RowAppend (obsolete), 890
RowDelete (obsolete), 890
RowDuplicate (obsolete), 891
RowInsert (obsolete), 891
RowMoveDown (obsolete), 892
RowMoveUp (obsolete), 892
SaveXML (obsolete), 893
SelectionMoveTabOrder (obsolete), 894
SelectionSet (obsolete), 895
XMLRoot (obsolete), 896
DOCTYPE Deklaration,
als Child in Grid-Ansicht hinzufügen, 414
in Grid-Anischt anhängen, 410
in Grid-Ansicht einfügen, 406
Konvertieren in (Grid-Ansicht), 417
Document,
Application, 756
AssignDTD, 757
AssignSchema, 757
AssignXSL, 757
AssignXSLFO, 757
AuthenticView, 758
Close, 758
ConvertDTDOrSchema, 759
CreateChild, 760
CreateSchemaDiagram, 760
CurrentViewMode, 760
DataRoot, 761
DocEditView, 761
Encoding, 761
EndChanges, 762
ExecuteXQuery, 762
ExportToDatabase, 762
ExportToText, 763
FullName, 764
GenerateDTDOrSchema, 764, 765
GenerateProgramCode, 765
GenerateSampleXML, 765
GenerateSchemaDocumentation, 766
GetExportElementList, 766
GetPathName, 767
GridView, 767
IsModified, 767
IsValid, 768
IsWellFormed, 769
Name, 769
OnBeforeCloseDocument, 754
OnBeforeSaveDocument, 754
OnBeforeValidate, 755
OnCloseDocument, 755
OnViewActivation, 756
Path, 770
RootElement, 770
Save, 770
SaveAs, 771
Saved, 771
SaveInString, 771
SaveToURL, 772
SetActiveDocument, 772
SetEncoding, 772
SetExternalIsValid, 774
SetPathName, 774
StartChanges, 774
SwitchViewMode, 775
Title, 775
TransformXSL, 776
TransformXSLFO, 776
UpdateViews, 776
UpdateXMLData, 776
XQuery, 762
Document-level,
examples of integration of XMLSpy, 946
integration of XMLSpy, 945, 946
Documents,
Count, 777
Item, 778

© 2007 Altova GmbH

Documents,
NewFile, 778
NewFileFromText, 778
OpenFile, 779
OpenURL, 779
OpenURLDialog, 780
Dokument,
Hinzufügen von Instanzen in Oracle XML DB, 522
Oracle XML DB durchsuchen, 520
Rechschreibprüfung, 559
Dokumentation,
für Altova-Produkte, 8
von WSDL-Dateien, 550
von XML-Schema, 439
Dokumente bearbeiten,
in der Grid-Ansicht, 138
Dokumentieren,
Schema, 52
Download Versionskontroll-Projekt, 385
Drag & Drop,
Elemente hinzufügen, 31
Draw menu,
in Scripting Environment, 644
Druckansicht, 367
Drucken,
von der Authentic-Ansicht, 94
Druckereinrichtung, 367
DTD, 587, 589
Attlist-Deklaration, 408, 412, 415, 417
Code generieren von, 437
Element-Deklaration, 408, 412, 415, 417
Entities inkludieren, 428
Entity-Deklaration, 408, 412, 415, 417
Entwurf anhand von XML-Datei erzeugen, 435
Gehe von XML-Dokument zu Definition, 429
Gehe von XML-Dokument zu DTD, 429
in UML konvertieren, 434
Konvertieren in XML-Schema, 431
Menübefehle zu, 428
Notation Deklaration, 417
Notation-Deklaration, 408, 412, 415
Referenz von XML zu externer DTD, 407
von XML-Dokument generieren, 429
zu XML-Dokument zuweisen, 428
DTD/Schema,
Menü, 428
DTDSchemaGeneratorDlg,
Application, 781
AttributeTypeDefinition, 781
DTDSchemaFormat, 781
FrequentElements, 781
GlobalAttributes, 781
MaxEnumLength, 782
MergeAllEqualNamed, 782
OnlyStringEnums, 782
OutputPath, 782
OutputPathDialogAction, 783
Parent, 783
ResolveEntities, 783
TypeDetection, 783
ValueList, 783
Durchsuchen,
Oracle XML DB, 520
Oracle XML DB abfragen, 516
Dynamische (SPS) Tabellen in der Authentic-Ansicht,
Verwendung, 202
Dynamische Tabellen,
bearbeiten, 191

E

Eclipse,
Dateien importieren, 241
starten und Projekt definieren, 237
und SOAP Debugger, 247
und XMLSpy, 236
und XSLT Debugger, 245
Unterschiede zu XMLSpy, 244
XML-Datei erstellen, 239
Edit menu,
in Scripting Environment, 639
Eigenschaften,
Oracle XML DB, 521
Versionskontrolle, 395
Einchecken, 389
Einfügen,
Befehl, 370
Breakpoint - XSLT Debugger, 462
in Grid-Ansicht, 405
Zeile (Dokument Editor), 470
Eingabehilfe, 602, 603
Details, 27
Eingabehilfen, 21
aktualisieren, 427

© 2007 Altova GmbH

Eingabehilfen, 21
Anzeige ein/aus, 603
für XQuery, 121
Grid-Ansicht, 134
in der Textansicht, 107
in Grid-Ansicht, 68
in Schema-Ansicht, 145
Eingabehilfen aktualisieren,
Befehl, 427
Eingabemethoden, 1023
Eingabetaste,
drücken, 219
Einklappen,
Child-Nodes ausklappen (Textansicht), 546
Einrücken,
in Text-Ansicht, 373
Einrücklinien,
in Textansicht, 106
Einsteigen,
XSLT Debugger, 461
Einstellungen, 587
Skripting, 600
XSLT Debugger, 464
Einstellungen für den Dateivergleich, 571
Element,
als Child in Grid-Ansicht hinzufügen, 413
in Grid-Ansicht anhängen, 409
in Grid-Ansicht einfügen, 405
Konvertieren in (Grid-Ansicht), 416
ELEMENT Deklaration,
als Child in Grid-Ansicht hinzufügen, 415
Konvertieren in (Grid-Ansicht), 417
ELEMENT-Deklaration,
in Grid-Ansicht anhängen, 412
ELEMENT-Deklaration in DTD,
in Grid-Ansicht einfügen, 408
Elemente,
Inhalt einschränken, 34
optional machen, 34
Elemente erweitern,
in Textansicht, 106
Elemente reduzieren,
in Textansicht, 106
Element-Eingabehilfe,
in der Authentic-Ansicht, 195
ElementFormDefault,
Einstellungen in der Schema Design-Ansicht, 438
ElementList,
Count, 784
Item, 784
RemoveElement, 784
ElementListItem,
ElementKind, 785
FieldCount, 785
Name, 785
RecordCount, 785
Elementtyp,
in XML-Dokument festlegen, 58
E-Mail,
Dateien senden, 365
Enable tracepoint - XSLT debugger, 463
Encoding, 362
in XQuery-Dokument, 995
Standard, 598
Endbenutzer-Lizenzvereinbarung, 1026, 1030
Enhanced Grid-Ansicht, 542
Beschreibung, 134
siehe Grid-Ansicht, 60
Entities,
in der Authentic-Ansicht definieren, 217
in der Authentic-Ansicht Entities einfügen, 93
Entities-Eingabehilfe,
in der Authentic.Ansicht, 195
ENTITY Deklaration,
als Child in Grid-Ansicht hinzufügen, 415
in Grid-Ansicht anhängen, 412
Konvertieren in (Grid-Ansicht), 417
Entity-Deklaration,
in Grid-Ansicht einfügen, 408
Entkoppeln,
Skripts von Projekt, 566
Enumerationen,
für Attribute definieren, 47
Enumerations,
in XMLSpyControl, 984
SPYAttributeTypeDefinition, 898
SPYAuthenticActions, 898
SPYAuthenticDocumentPosition, 898
SpyAuthenticElementActions, 898
SPYAuthenticElementKind, 899
SPYAuthenticMarkupVisibility, 899
SPYDatabaseKind, 899
SPYDialogAction, 899
SPYDOMType, 900
SPYDTDSchemaFormat, 900
SPYEncodingByteOrder, 900

© 2007 Altova GmbH

Enumerations,
SPYExportNamespace, 900
SPYFrequentElements, 900
SPYKeyEvent, 901
SPYLibType, 901
SPYLoading, 901
SPYMouseEvent, 901
SPYNumberDateTimeFormat, 902
SPYProgrammingLanguage, 902
SPYProjectItemTypes, 902
SPYProjectType, 903
SPYSampleXMLGenerationOptimization, 903
SPYSampleXMLGenerationSchemaOrDTDAssignment, 903
SPYSchemaDefKind, 903
SPYSchemaDocumentationFormat, 904
SPYTextDelimiters, 904
SPYTextEnclosing, 904
SPYTypeDetection, 904
SPYURLTapes, 905
SPYViewModes, 905
SPYVirtualKeyMask, 905
SPYXMLDataKind, 906
Erneut starten,
XSLT Debugger, 461
Ersetzen,
Text, 373
Text in Dokument, 377
Text in mehreren Dateien, 378
Erstellen,
Neues XML Dokument, 56
XML-Datei in Eclipse, 239
Erweitern,
alles, 544
Child-Nodes einklappen (Textansicht), 546
Erweiterte Validierung, 177
in SchemaAgent, 451
Erzeugen,
DB auf Basis von Schema, 534
Evaluierungs-Keycode,
für Ihre Altova-Software, 606
Evaluierungszeitraum,
bei Altova Software-Produkten, 1029
für Altova-Software-Produkte, 1027
von Altova Software-Produkten, 1026
Event, 685, 686, 687, 727, 728, 730, 731, 732, 734, 754, 755, 756, 804, 805, 806
Event Handlers,
in Scripting Environment, 627
invoking, 615
Events, 667
Exception,
out of memory, 334
Explorer, 589
ExportSettings,
CreateKeys, 786
ElementList, 786
EntitiesToText, 787
ExportAllElements, 787
FromAttributes, 787
FromSingleSubElements, 787
FromTextValues, 787
IndependentPrimaryKey, 788
Namespace, 788
SubLevelLimit, 788
Extern geparste Entitäten, 589
ExternalID Deklaration,
als Child in Grid-Ansicht hinzufügen, 414
in Grid-Ansicht anhängen, 411
in Grid-Ansicht einfügen, 407
Konvertieren in (Grid-Ansicht), 417
Externe Funktionen,
in XQuery-Dokument, 995
Externer XSL Prozessor, 599
Extras,
Menü, 559

F

FAQs,
auf der Altova Website, 8
Farbe, 593, 595
Farben,
Register, 596
Tabelle, 596
Favoriten, 604
Fehlerbehebung,
Online-Hilfe und Support, 8
Fenster,
Eingabehilfe, 602, 603
horizontal nebeneinander, 602
Info, 602, 603
Menü, 602
öffnen, 603

© 2007 Altova GmbH

Fenster,
Projekt, 602, 603
überlappend, 602
Übersicht, 21
vertikal anordnen, 602
Fenster "Trace", 274
Fenster Trace,
im XSLT/XQuery Debugger, 268
File menu,
in Scripting Environment, 639
FileSelectionDlg,
Application, 789
DialogAction, 789
FullName, 789
Parent, 790
Find in Files output window, 111
Finden,
siehe Suchen, 377
fn:base-uri in XPath 2.0,
Unterstützung in Altova-Prozessoren, 999
fn:collection in XPath 2.0,
Unterstützung in Altova-Prozessoren, 999
fn:count() in XPath 2.0,
und Whitespace, 998
fn:current-date in XPath 2.0,
Unterstützung in Altova-Prozessoren, 999
fn:current-dateTime in XPath 2.0,
Unterstützung in Altova-Prozessoren, 999
fn:current-time in XPath 2.0,
Unterstützung in Altova-Prozessoren, 999
fn:data in XPath 2.0,
Unterstützung in Altova-Prozessoren, 999
fn:deep-equal() in XPath 2.0,
und Whitespace, 998
fn:id in XPath 2.0,
Unterstützung in Altova-Prozessoren, 999
fn:idref in XPath 2.0,
Unterstützung in Altova-Prozessoren, 999
fn:index-of in XPath 2.0,
Unterstützung in Altova-Prozessoren, 999
fn:in-scope-prefixes in XPath 2.0,
Unterstützung in Altova-Prozessoren, 999
fn:last() in XPath 2.0,
und Whitespace, 998
fn:lower-case in XPath 2.0,
Unterstützung in Altova-Prozessoren, 999
fn:normalize-unicode in XPath 2.0,
Unterstützung in Altova-Prozessoren, 999
fn:position() in XPath 2.0,
und Whitespace, 998
fn:resolve-uri in XPath 2.0,
Unterstützung in Altova-Prozessoren, 999
fn:static-base-uri in XPath 2.0,
Unterstützung in Altova-Prozessoren, 999
fn:upper-case in XPath 2.0,
Unterstützung in Altova-Prozessoren, 999
Foren,
zur Diskussion von XML-Problemen, 8
Form Editor,
description of, 622
Form Objects,
in Scripting Environment, 621
Forms,
creating in Scripting Environment, 629
designing with Form Editor, 622
in scripting projects, 614
running in application, 615
functions,
XPath 2.0 and XQuery 1.0, 998
Funktionen,
siehe unter XSLT 2.0-Funktionen, 994
Funktionstasten, 577

G

Gehe zu Datei, 545
Gehe zu Zeile/Zeichen, 545
Gemischte Markups,
anzeigen, 470
Generate,
code from schema, 303
Generate Sample XML, 898, 903
GenerateSampleXMLDlg,
Application, 800
FillWithSampleData, 802
NonMandatoryAttributes, 801
NonMandatoryElements, 801
Parent, 801
RepeatCount, 801
TakeFirstChoice, 801
Generieren,
DB-Struktur auf Basis von Schema, 534
Global, 1023
Global objects,

© 2007 Altova GmbH

Global objects,
in SPL, 340
Globale,
Einstellungen, 587
Globale Elemente,
in XML-Schemas verwenden, 45
Globale Elemente anzeigen, 445
Globale Komponenten,
Auflistung in GUI, 145
in der Schema-Übersicht erstellen, 150
gradient,
background, 336
Grafikformate,
in der Authentic-Ansicht, 219
Grammatik, 589
Graue Leiste, 543, 544
Grid View,
Datenbank/Tabellen-Ansicht, 140
Grid-Ansicht, 542, 544
Attribut / Element anhängen, 68
bearbeiten, 420
Beschreibung, 134
Dateneingaben, 60
editieren in, 138
Eingabehilfen, 134
Eingabehilfen verwenden, 68
und Tabellenansicht, 70
zu Tabellenansicht wechseln, 418
Grid-Schriftarten, 593
GridView,
CurrentFocus, 806
Deselect, 806
IsVisible, 807
OnBeforeDrag, 804
OnBeforeDrop, 804
OnBeforeStartEditing, 805
OnEditingFinished, 805
OnFocusChanged, 806
Select, 807
SetFocus, 807
Größe, 595
Große Markups,
anzeigen, 470
Große Markup-Symbole anzeigen, 191, 193
GUI,
Übersicht, 14

H

Hauptfenster, 21
Hebräisch, 1024
Hilfe,
Index, 604
Inhaltsverzeichnis, 604
Menü, 604
Suchen, 604
Tastaturzuordnungen, 605
Hilfesystem, 604
Hin- und Herschalten, 602, 603
Hintergrundinformationen, 1019
Hin-und Herschalten, 602
Hinzufügen,
Dateien zu Projekt, 396
Ordner zu Projekt, 397
Projekt in Versionskontrolle, 391
Versionskontrolle, 391
Hiragana, 1023
Höher stufen,
Befehl, 420
HTML,
integration of XMLSpy, 949, 950, 951, 953
HTML example,
of XMLSpyControl integration, 942, 943, 944

I

IBM DB2,
Schemaverwaltung und -zuweisung, 495
Verwalten von XML-Schemas, 495
XML Schemas einer XML-Datei zuweisen, 499
Identity Constraint,
in der Content Model-Ansicht ein- und ausblenden, 47
Identity Constraints,
in der Schema-Übersicht für eine Schema-Komponente definieren, 150
Implementierungsspezifisches Verhalten,
von XSLT 2.0.Funktionen, 994
Implizite Zeitzone,
und XPath 2.0-Funktionen, 998
Import,
Datenbankdaten basierend auf XML-Schema, 533

© 2007 Altova GmbH

Import,
MS Word Dokument als XML, 529
Importieren,
Datenbankstruktur als XML-Schema, 529
Textdatei als XML, 523
In Dateien suchen (Befehl), 378
In Element einbinden,
Befehl, 420
Index,
Hilfe, 604
Info,
Fenster, 21
Infofenster, 21
Info-Fenster, 602, 603
Info-Fenster in XSLT/XQuery Debugger, 268
Informationsfenster,
anordnen, 268
Inhaltsverzeichnis, 604
Insert,
tracepoint - XSLT debugger, 462
Instanzen,
Anzahl, 27
Integrating,
XMLSpy in applications, 941
Intelligentes Editieren, 591
in der Textansicht, 117
Internet, 607
Internet-Verwendung,
in Altova-Produkten, 1025
Introduction,
code generator, 304
ISO/IEC 10646, 1023

J

JAVA,
API, 907
ClassPath, 907
code, 303
settings, 336

K

Kanji, 1023
Kapitel, 604
Katakana, 1023
Katalog,
Oasis XML, 422
Keycodes,
für Ihre Altova-Software, 606
Klappleiste,
in Textansicht, 106
Kleine Markups,
anzeigen, 470
Kleine Markup-Symbole anzeigen, 191, 193
Kodierung,
von Dateien, 362
Kommentare in SQL-Scripts, 489
Kommunikationsvorgang,
SOAP Debugger, 293
Komponenten,
Auflistung in GUI, 145
Komponenten-Definitionen,
wiederverwenden, 38
Komponenten-Navigator,
Beschreibung, 145
Kompositor,
für Sequences, 27
Kompositoren,
in der Schema-Ansicht, 154
Kontextfenster,
im XSLT/XQuery Debugger, 265
Kontextmenü,
anpassen, 585
Befehle, 581
Kontextmenüs,
in der Authentic-Ansicht, 195, 199
Konvertieren,
Datenbankschema in XML-Schema, 529
Menü, 523
MS Word Daten in XML, 529
Oracle XML DB, 516
Schema in DB-Struktur, 534
Textdateo in XML, 523
Konvertieren in,
in Grid-Ansicht, 416
Kopieren,
als strukturierten Text, 370
als XML-Text, 370
Befehl, 369
XPath, 373
Zeile (Dokument Editor), 471

© 2007 Altova GmbH

Kürzel, 577
anzeigen in Tooltip, 585
zuweisen/löschen, 577

L

last() function in XPath 2.0,
siehe fn:last(), 998
last()-Funktion,
in XPath 1.0, 990
Layers,
in design of Forms with Form Editor, 622
Layout menu,
in Scripting Environment, 641
Layout of Forms,
in Form Editor, 622
Leere Elemente, 589
Lesenzeichen,
navigieren, 380
Lesezeichen, 604
alle löschen, 380
einfügen und löschen, 380
in Textansicht, 106
navigieren, 380
Lesezeichen in SQL-Scripts, 489
Lesezeichenleiste,
in Textansicht, 546
Lib,
compiler settings, 336
Library, 348
Little-endian, 598
Lizenz, 1030
für Ihre Altova-Software, 606
Informationen, 1026
Lizenzüberwachung,
bei Altova-Produkten, 1028
loading, 779
Lokales Projekt, 385
Löschen,
Befehl, 370
Befehl aus Symbolleiste, 574
Befehl im Kontextmenü, 581
Breakpoint - XSLT Debugger, 462
Kürzel zu Befehl, 577
Symbol aus Symbolleiste, 574
Symbolleiste, 575
Zeile (Dokument Editor), 471

M

Macros,
creating in Scripting Environment, 634
how to call, 636
running in application, 615
Main Window,
in Scripting Environment, 620
MainCatalog, 422
Makro,
Hinzufügen zu Menü/Symbolleiste, 583
Schaltfläche bearbeiten, 585
Makros anzeigen,
Skripting, 565
MapForce, 435
Markup,
ausblenden, 470
Gemischte anzeigen, 470
in der Authentic-Ansicht, 191
kleine anzeigen, 470
Markups,
große anzeigen, 470
Markup-Symbole,
in der Authentic-Ansicht, 193
Markup-Symbole ausblenden, 191, 193
Maximale Zellenbreite, 592
Memory,
out of exceptions, 334
Menü,
anpassen, 581
Ansicht, 542
Bearbeiten, 369
Befehl hinzufügen/löschen, 574
Befehle löschen, 581
Browser, 547
Datei, 356
Extras, 559
Fenster, 602
Hilfe, 604
Konvertieren, 523
Makro hinzufügen, 583
Projekt, 382
SOAP, 552
Standard/XMLSPY, 581

© 2007 Altova GmbH

Menü,
WSDL, 548
Messages-Fenster,
in XSLT/XQuery Debugger, 267
Method,
Reserve name, 334
MIME, 589
Mixed Markup-Symbole anzeigen, 191, 193
Mode,
compatible to v2005R3, 336
Modules,
in Scripting Environment, 620
in scripting projects, 614
MS Access,
Datentypen von XML-Schema, 1011
Konvertierung von Datentypen in XML-Schema Generierung von DB, 1003
MS Office Open XML-Dokumente,
bearbeiten in XMLSpy, 224
MS SQL Server,
Datentypen von XML-Schema, 1013
Konvertierung von Datentypen in XML-Schema Generierung von DB, 1004
Schema Extensions, 446, 447
Schema-Einstellungen, 448
MS Visual Source Safe, 383
MSXML, 599
Multi-user, 587
MySQL,
Datentypen von XML-Schema, 1015
Konvertierung von Datentypen in XML-Schema Generierung von DB, 1005

N

Nach oben,
Zeile verschieben (Dokument Editor), 471
Nach unten,
Zeile verschieben (Dokument Editor), 471
Named Schema-Beziehungen,
MS SQL Server Schema-Einstellungen, 447
Namepsaces,
in WSDL-Dokumenten, 96
Namespace Präfix,
in Grid-Ansicht einfügen, 427
Namespaces,
Einstellungen in der Schema Design-Ansicht, 438
im XSLT 2.0 Stylesheet, 992
in Schemas, 26
in XQuery-Dokument, 995
Native Oberfläche ausführen, 396
Navigieren,
Kürzel in Schema, 50
Nebeneinander, 592
horizontal, 602
vertikal, 602
Neu laden, 587
geänderte Dateien, 361
Neue Datei,
erstellen, 356
Neues XML-Dokument,
erstellen, 56
Nicht lateinischen Schriftsysteme, 1023
Nicht-XML Dateien, 589
Nillable,
not supported, 334
Node,
aktuellen Execution Node einblenden, 462
NOTATION Deklaration,
als Child in Grid-Ansicht hinzufügen, 415
in Grid-Ansicht anhängen, 412
in Grid-Ansicht einfügen, 408
Konvertieren in (Grid-Ansicht), 417

O

OASIS,
XML-Katalog, 422
Objektsuche,
im Fenster "Datenbankabfrage", 485
ODBC,
Konvertierung von Datentypen in XML-Schema Generierung von DB, 1007
ODBC-Verbindungen, 480
Office Open XML,
siehe MS Office Open XML, 224
Öffnen, 359
Projekt-Versionskontrolle, 385
Online-Hilfe,
für Altova-Produkte, 8
Open XML,
siehe MS Office Open XML, 224
Optimale Breite, 544, 592

© 2007 Altova GmbH

Optionale Elemente,
erstellen, 34
Optionen beim Öffnen,
Datei, 587
Oracle,
Datentypen von XML-Schema, 1017
Konvertierung von Datentypen in XML-Schema Generierung von DB, 1006
Schema-Extensions, 445
Oracle XML Datenbank, 516
Oracle XML DB,
Datenbank durchsuchen, 520
DB durchsuchen, 516
Dokumentinstanzen hinzufügen, 522
Eigenschaften, 521
Schema auflisten, 518
Schema hinzufügen, 519
Schema öffnen, 518
Oracle XML DBUri Statements, 516
Ordner,
Dateien in Eclipse importieren, 241
OrgChart.sps,
im Tutorial der Authentic-Ansicht, 83
OrgChart.xml,
im Tutorial der Authentic-Ansicht, 83
OrgChart.xsd,
im Tutorial der Authentic-Ansicht, 83
Out of memory, 334
Overview,
of XMLSpy API, 663

P

Parameter,
in DB-Abfragen, 210
Übergabe an Stylesheet über Schnittstelle, 455
Parent, 770
Parser,
in Altova-Produkte integrierter, 1021
XSLT, 599
Perspektive,
XMLSpy in Eclipse, 250
Plattformen für Altova-Produkte, 1020
Plug-in, 584
ATL sample files, 652
registration, 647
User interface configuration, 649
XMLSPY, 646
Position,
Zeichen, 545
Zeile, 545
position() function,
in XPath 1.0, 990
position() function in XPath 2.0,
siehe fn:position(), 998
Pretty-Print,
Text-Ansicht, 373
PRJ files,
for Scripting Environment, 614
Processing Instruction,
als Child in Grid-Ansicht hinzufügen, 414
in Grid-Ansicht anhängen, 410
in Grid-Ansicht einfügen, 406
Konvertieren in (Grid-Ansicht), 417
Profiler, 279, 459
Profilerstellung,
Callgraph, 279
Trefferquote, 279
Programmeinstellungen, 587
Programming points,
in Scripting Environment, 638
Project menu,
in Scripting Environment, 640
Project Window,
in Scripting Environment, 618
Projekt,
aktive Dateien hinzufügen, 397
Dateien zu Projekt hinzufügen, 396
Eigenschaften, 402
entfernen aus Versionskontrolle, 392
externe Ordner hinzufügen, 397
externen Webordner hinzufügen, 400
Hinzufügen aktiver und verwandter Dateien, 397
Hinzufügen in Versionskontrolle, 391
in Eclipse definieren, 237
Menü, 382
Neu, 383
Neu laden, 383
öffnen, 383
Ordner hinzufügen, 397
schließen, 383
Skripting, 566
Skripts aktivieren, 566
Skripts entkoppeln, 566

© 2007 Altova GmbH

Projekt,
Speichern, 383
URL zu Projekt hinzufügen, 396
zuletzt verwendete, 404
zuweisen zu Skript, 566
Projekt öffnen,
Versionskontrolle, 385
Projekte,
Übersicht, 382
Projekte in XMLSpy,
erstellen, 81
Vorteile, 80
Projektfenster, 21
Anzeige ein/aus, 603
Projekt-Fenster, 602, 603
Projektskripts aktivieren, 566
Projektverwaltung in XMLSpy, 80
Property Sheet,
in Form Editor, 622
Provider,
auswählen, 385
Prozessor-Informationen, 989
PUBLIC,
Identifier - Katalog, 422
PVCS Version Manager, 383

Q

QName Serialisierung,
Rückgabe durch XPath 2.0 -Funktionen, 999

R

Rechschreibprüfung, 559
benutzerdefiniertes Wörterbuch, 559
Rechtliches, 1026
Rechtschreibprüfung,
Optionen, 561
Reduzieren,
unselektierte reduzieren, 544
Referenz, 355
Referenz für Programmierer, 610
Regionen in SQL-Scripts, 489
Register, 587
plug-in, 647
Registrieren,
Schema - Oracle XML DB, 519
Registrierung der Altova-Software, 606
Registry,
Einstellungen, 587
Regular Expression,
in Suchstring, 373
Regular Expressions, 378
Release notes,
XMLSpy API, 611
Remove,
tracepoint - XSLT debugger, 462
Replace, 111
Request,
SOAP Request senden, 553
von WSDL senden, 102
Reserve,
method name, 334
RichEdit 3.0, 595
Rückgängig, 369
Rückgängig-Befehl, 117
Rückwärtskompatibilität,
des XSLT 2.0-Prozessors, 992

S

save, 772
schema, 693
code generator, 303
DB auf Basis von Schema erzeugen, 534
Design Ansicht, 543
Dokumentation, 52
Dokumentation Schriftart, 593
einer DB-XML zuweisen, 499
Einstellungen, 587
Einstellungen für Oracle, 446
In Oracle XML DB registrieren, 519
in UML konvertieren, 434
Schema von Datenbank öffnen, 518
siehe auch XML-Schema, 22, 428
Verwaltung und Zuweisung in IBM DB2-Datenbanken, 495
WSDL Schema öffnen, 101
Schema auflisten,
Oracle XML DB, 518
Schema Design,

© 2007 Altova GmbH

Schema Design,
Menü, 438
Schema Design-Ansicht,
Beschreibung, 145
Diagramm anzeigen, 445
Globale Elemente anzeigen, 445
Zoom-Funktion, 444
Schema hinzufügen,
Oracle XML DB, 519
Schema Schriftarten, 593
Schema/WSDL Design-Ansicht,
Beschreibung, 144
SchemaAgent,
arbeiten mit, 172, 173, 177
erweiterte Validierung, 451
Schemas anzeigen, 450
Schemas von XMLSpy aus öffnen, 177
Verbindung trennen, 450
Verbindung zu Server herstellen von XMLSpy, 449
SchemaAgent in XMLSpy, 168
SchemaAgent Server,
Verbindung herstellen, 170
Schema-Ansicht,
Eingabehilfen in, 145
konfigurieren, 32
SchemaDocumentationDlg,
AllDetails, 808
Application, 808
IncludeAll, 809
IncludeAttributeGroups, 810
IncludeComplexTypes, 810
IncludeGlobalElements, 810
IncludeGroups, 811
IncludeIndex, 811
IncludeLocalElements, 811
IncludeRedefines, 812
IncludeSimpleTypes, 812
OptionsDialogAction, 813
OutputFile, 813
OutputFileDialogAction, 813
OutputFormat, 814
Parent, 814
ShowAnnotations, 814
ShowAttributes, 815
ShowChildren, 815
ShowConstraints, 816
ShowDiagram, 815
ShowEnumerations, 815
ShowNamespace, 816
ShowPatterns, 816
ShowProgressBar, 817
ShowProperties, 817
ShowResult, 817
ShowSingleFacets, 817
ShowSourceCode, 818
ShowType, 818
ShowUsedBy, 818
Schemafähigkeit,
des XPath 2.0- und XQuery-Prozessors, 998
schemanativetype, 338
Schemas,
für IBM DB2 verwalten, 495
im Speicher, 437
in Schema-Ansicht erstellen, 144, 145
Schema-Übersicht, 23
ausführliche Beschreibung, 150
Grundzüge, 145
Schema-Validierung eines XML-Dokuments,
für XQuery, 995
Schließen, 362
Projekt, 383
Schriftart, 1023
Schema, 593
Schema Dokumentation, 593
Schriftgröße,
in Browser-Ansicht, 547
Schriftsystem, 1023
Schriftsysteme, 1024
Script editing,
in Form Editor, 622
Scripting,
Skript zu Projekt zuweisen, 566
Scripting Environment,
documentation, 613
Form Editor, 622
Form Objects, 621
how to use, 627
launching, 618
Main Window, 620
programming points, 638
Project Window, 618
settings, 617
Scripting Environment. GUI description, 618
Scripting projects, 614
Sequence-Kompositor,
verwenden, 27

Settings,
c++ and c#, 336
SimpleTypes,
in Schema-Definitionen, 38
Skripting,
Makros anzeigen, 565
Projekt, 566
Projektskripts aktivieren, 566
Skripts aus Projekt löschen, 566
Zu Skripting-Umgebung wechseln, 565
Skripts, 600
Skriptsprache, 600
Smart Restrictions, 164
SOAP,
Debugger Optionen, 557
Debugger Sitzung, 556
Menü, 552
Neuen Request erstellen, 552
Proxyserver anhalten, 557
Proxyserver starten, 557
Request an Server senden, 553
Request Parameter, 555
Request von WSDL senden, 102
Requests, 557
Response, 557
SOAP Debugger,
Anwendungsbeispiel, 294
Beschreibung, 292
Breakpoints setzen, 300
in Visual Studio .NET, 234
SOAP Kommunikationsvorgang, 293
Software-Produktlizenz, 1030
Speicher, 383
Speichern von Schemainformationen, 437
Speichern,
alle, 364
Projekt, 383
SPL, 337
code blocks, 337
conditions, 342
foreach, 343
global objects, 340
subroutines, 344
using files, 341
variables, 339
Splashscreen, 592
SPP Datei-Speicherpfade, 382
SPS,
neuer XML-Datei zuweisen, 356
SPS-Datei,
zu XML-Datei zuweisen, 467
SPS-Tabellen,
dynamischen Tabellen bearbeiten, 191
SPS-Tabellen in der Authentic-Ansicht,
Verwendung, 202
SpyProject,
CloseProject, 819
ProjectFile, 819
RootItems, 819
SaveProject, 820
SaveProjectAs, 820
SpyProjectItem,
ChildItems, 820
FileExtensions, 821
ItemType, 821
Name, 821
Open, 821
ParentItem, 821
Path, 822
ValidateWith, 822
XMLForXSLTransformation, 822
XSLForXMLTransformation, 822
XSLTransformationFileExtension, 822
XSLTransformationFolder, 822
SpyProjectItems,
AddFile, 823
AddFolder, 823
AddURL, 823
Count, 824
Item, 824
RemoveItem, 824
SQL Editor,
Abfrage erstellen, 492
Beschreibung, 489
im Fenster Datenbankabfrage, 489
Standard,
Encoding, 598
Menü, 581
Standardansicht,
Einstellung im Hauptfenster, 589
Standardmäßiger Editor, 589
Star Team, 383
Start,
Eclipse, 237
XSLT Debugger, 461
Startgruppe,

© 2007 Altova GmbH

Startgruppe,
hinzufügen (Kontextmenü), 585
Statische (SPS) Tabellen in der Authentic-Ansicht,
Verwendung, 202
Status aktualisieren,
Versionskontrolle, 396
Stil, 593, 595
Stoppen,
XSLT Debugger, 461
Strukturierter Text, 591
Studio,
XMLSPY Integration in Visual Studio .NET, 226
Stylesheet PI, 459, 460
StyleVision, 435
zum Bearbeiten eines StyleVision Power Stylesheet, 468
StyleVision Power Stylesheet,
in StyleVision bearbeiten, 468
zu XML-Datei zuweisen, 467
StyleVision Power Stylesheet zu XML-Datei zuweisen, 467
Suchen,
Hilfe, 604
Text in Dokument, 373
und Text in Dokument ersetzen, 377
Support,
für Altova-Produkte, 8
Support Center, 607
Sybase,
Konvertierung von Datentypen in XML-Schema Generierung von DB, 1009
Symbol,
Hinzufügen zu Symbolleiste/Menü, 574
Symbole,
große anzeigen, 585
Symbolleiste,
Befehle anpassen, 574
Ein/Aus, 575
große Symbole anzeigen, 585
Makro hinzufügen, 583
Neue erstellen, 575
Symbolleiste & Menübefehle zurücksetzen, 575
Syntaxfärbung, 589, 592
für XQuery, 122

T

Tab sequence,
in Forms, 622
Tabelle,
automatisch erzeugen, 589
Befehl in Grid-Ansciht, 418
Farben, 596
Tabelle Ansicht, 591
Tabellen,
dynamische (SPS)-Tabellen (SPS) bearbeiten, 191
in der Authentic-Ansicht, 87
in und aus externen Applikationen exportieren und importieren, 140
Tabellen in der Authentic-Ansicht,
Arbeiten mit (statischen und dynamischen) SPS Tabellen, 202
Arbeiten mit XML-Tabellen, 203
Symbole zum Bearbeiten von XML-Tabellen, 207
Verwendung, 202
Tabellenansicht,
in Grid-Ansicht wechseln, 418
Spalten sortieren, 419
Verwendung, 70
Zeile anhängen, 419
Zeile einfügen, 419
Tabellenansicht in der Enhanced Grid-Ansicht, 140
Tamino,
Schema-Einstellungen, 449
Schema-Extensions, 448
Tamino in XMLSPY,
Beschreibung, 500
Tastaturzuordnungen, 605
Tatstaturkürzel, 577
Technische Informationen, 1019
Technischer Kundendienst, 607
Technischer Support,
für Altova-Produkte, 8
Templates-Fenster,
in XSLT/XQuery Debugger, 267
terminate, 696
Text,
als Child in Grid-Ansicht hinzufügen, 413
in Dokument suchen, 373
in Grid-Ansicht anhängen, 409
in Grid-Ansicht einfügen, 405

© 2007 Altova GmbH

Text,
Konvertieren in (Grid-Ansicht), 416
Nodes ein-/ausklappen, 546
Pretty-Print, 373
Schriftart, 595
suchen und ersetzen, 377
Textansicht, 542
Bearbeiten in, 60
Beschreibung, 105
Eingabehilfen, 107
intelligentes Editieren, 117
Lesezeichenleiste, 546
Text-Ansicht,
Auto-Komplettierung Ein/Aus, 591
Schema Schriftarten, 593
Textdatei,
als XML importieren, 523
Export von XML-Daten, 536
TextImportExportSettings,
DestinationFolder, 825
EnclosingCharacter, 825
Encoding, 825
EncodingByteOrder, 826
FieldDelimiter, 826
FileExtension, 826
HeaderRow, 826
ImportFile, 826
Themen,
Ansicht im Inhaltsverzeichnis, 604
Tiefer stufen,
Befehl, 420
Tooltip,
anzeigen, 585
Kürzel anzeigen, 585
Tracepoints,
in XSLT/XQuery Debugger verwenden, 274
Transformation,
siehe XSLT-Transformation, 454
Trefferquote-Profilerstellung, 279
Tutorial,
Beispieldateien, 21
WSDL, 96
Ziele, 21
type,
Erweiterung im XML-Dokument, 58
Types,
built in, 347

U

Überlappend,
Fenster, 602
Übernehmen, 587
Überprüfen,
Rechtschreibung, 559
Übersicht, 21
Überspringen,
XSLT Debugger, 462
UCS-2, 598, 1023
UCS-4, 1023
UML,
Schemas konvertieren, 434
Unicode,
Konsortium, 1023
Standard, 1023
Unterstützung, 1023
Unicode-Zeichen, 1023
Universalschriftart, 1023
Unnamed Element Relationships,
MS SQL Server Schema-Einstellungen, 448
Unselektierte, 544
Unterschiede anzeigen, 394
Urheberrecht, 607
URL, 698, 699, 772, 779, 780
Datei öffnen, 359
Datei speichern unter, 363
per E-Mail senden, 365
User interface,
configure using plug-in, 649
UTF-16, 598
UTF-8, 1023

V

Validator,
in Altova-Produkten, 1021
Validieren, 422
Ausgabefenster der Validierung, 110
WSDL-Datei, 101
XML-Dokument, 64
Validierung,

© 2007 Altova GmbH

Validierung,
DTD zu XML-Dokument zuweisen, 428
von verknüpften Schemas mit Hilfe von SchemaAgent, 177
WSDL-Dateien, 426
XML-Schema zu XML-Dokument zuweisen, 428
Variablenfenster,
in XSLT/XQuery Debugger, 265
Variables,
in SPL, 339
Verbergen, 602, 603
Verbindung mit SchemaAgent Server trennen, 450
Verbindung zu einer Datenbank herstellen, 472
Verbindung zu SchemaAgent Server herstellen, 170
Verbindung zu SchemaAgent Server von XMLSpy, 449
Verbindung zur Datenquelle, 480
Verbindung zur Datenquelle herstellen, 473, 476
Verbindungsassistent, 473
Vergrößern/Verkleinern,
in Schema Design-Ansicht, 444
Verlauf anzeigen, 393
Version changes,
XMLSpy API, 611
Versionsdateien vergleichen, 394
Versionskontrolle, 383
aktivieren, 387
aktuellste Version holen, 388
auschecken, 388
Auschecken rückgängig, 390
ausgliedern, 392
deaktivieren, 387
Eigenschaften, 395
Einchecken, 389
Hinzufügen in Versionskontrolle, 391
native Oberfläche ausführen, 396
Projekt öffnen, 385
Status aktualisieren, 396
Unterschiede anzeigen, 394
Verlauf anzeigen, 393
Versionsnummer, 607
Vertrieb,
von Altova Produken, 1029
von Altova Software-Produkten, 1026
von Altova-Software-Produkten, 1027
Verzeichnisse,
zwei vergleichen, 569
Verzeichnisse vergleichen, 569
View menu,
in Scripting Environment, 639
Visual Studio .Net,
und SOAP Debugger, 234
und XMLSPY, 226
und XMLSPY Unterschiede, 227
und XSLT Debugger, 231
Volltextsuche, 604
Von rechts nach links, 1024
Vorlagen,
Ordner, 21
von XML-Dokumenten in Authentic-Ansicht, 466
Vorlagendateien,
für neue Dokumente, 356

W

Web Server, 607
Webservice,
Verbindung herstellen, 101
Wechseln,
Zu Skript-Umgebung, 565
Weitersuchen,
Befehl, 377
Whitespace im XML-Dokument,
Behandlung durch den Altova XSLT 2.0-Prozessor, 992
Whitespace Nodes im XML-Dokument,
und Behandlung durch den XSLT 1.0-Prozessor, 990
Whitespace-Behandlung,
und XPath 2.0-Funktionen, 998
Wiedergabe, 1024
Wiederherstellen,
Befehl, 369
Wiederholende Elemente, 591
Window menu,
in Scripting Environment, 645
Windows,
Unterstützung für Altova-Produkte, 1020
Windows 2000, 1023
Windows NT, 1023
Wohlgeformtheitsprüfung, 421
bei XML-Dokument, 64
Word-Dokument,
als XML importieren, 529
Workbench,
Eclipse, 237
Wörterbuch,
aktuelles ändern, 559

© 2007 Altova GmbH

Wörterbuch,
benutzerdefiniertes Wörterbuch hinzufügen, 559
Rechschreibprüfung, 559
WSDL,
Binding, 549
Bindings erstellen, 99
creating messages, 97
creating operations, 97
creating parameters, 97
creating PortTypes, 97
Dokumentation erstellen, 103
Einstellungen, 551
Erzeugen von Dokumentation, 550
Messages, 548
Namespaces, 96
neues Dokument erstellen, 96
Operations, 548
Ports erstellen, 100
PortType, 548
Schema öffnen, 101
Service, 549
Services erstellen, 100
SOAP Debugger-Beispiel, 294
SOAP Request senden, 102
Tutorial, 96
Types, 549
validieren, 101
Verbindung zu einem Webservice herstellen, 101
Web service, 552
WSDL Design-Ansicht,
Beschreibung, 180
Bindings, 182
Dateien anzeigen, 180
Details-Eingabehilfe, 185
Eingabehilfen, 185
Funktionalitäten, 180
Hauptfenster, 182
importieren in ein WSDL-Dokument, 185
PortTypes, 182
Services, 182
Übersicht-Eingabehilfe, 185
WSDL Menü, 548
WSDL-Dateien,
Validierung, 426
WSDL-Schriftarten, 594

X

Xerces,
support, 336
XML, 1023
Dokument, 55
Hinzufügen von Dokumentinstanzen in Datenbank, 522
in Eclipse erstellen, 239
in Eclipse importieren, 241
Menü, 405
Neues Dokument, 56
Oasis-Katalog, 422
Rechschreibprüfung, 559
Symbole, 55
XML Diff,
Dateien vergleichen, 571
Dateivergleich, 566
Verzeichnisse vergleichen, 569
XML Dokument,
Oracle XML DB durchsuchen, 520
XML Import,
Basierend auf Schema, 533
XML Parser,
Info, 1021
XML Schema,
MS SQL Server Schema-Einstellungen, 448
XML; Kopieren als strukturierten Text, 370
XMLData,
AppendChild, 842
EraseAllChildren, 843
EraseCurrentChild, 844
GetChild, 844
GetChildKind, 845
GetCurrentChild, 845
GetFirstChild, 845
GetNextChild, 846
HasChildren, 847
HasChildrenKind, 847
InsertChild, 848
IsSameNode, 848
Kind, 848
MayHaveChildren, 848
Name, 849
Parent, 849
TextValue, 849

© 2007 Altova GmbH

XML-Datei,
von DTD oder XML-Schema erzeugen, 435
XML-Daten,
in Datenbank exportieren, 539
in Textdatei exportieren, 536
XML-Daten in Datenbank exportieren, 539
XML-Daten in Textdateien exportieren, 536
XML-Dokument,
in der Authentic-Ansicht öffnen, 84
in Textansicht bearbeiten, 60
to XSLT Stylesheet zuweisen, 460
XML-Konformität, 589
XML-Prolog,
als Child in Grid-Ansicht hinzufügen, 414
in Grid-Ansicht anhängen, 410
in Grid-Ansicht einfügen, 406
Konvertieren in (Grid-Ansicht), 417
XML-Schema,
Ansicht konfigurieren, 32
Code generieren von, 437
Content Model-Ansicht konfigurieren, 441
Content Model-Diagramm, 439
Dokumentation generieren, 439
einer DB-XML zuweisen, 499
Einstellungen in der Schema Design-Ansicht, 438
Erstellen eines einfachen Schemas, 22
für IBM DB2 verwalten, 495
Gehe von XML-Dokument zu Definition, 429
Gehe von XML-Dokument zu Schema, 429
in DTD konvertieren, 431
Komponenten hinzufügen, 27
Menübefehle zu, 428
MS SQL Server Extensions, 446
MS SQL Server Schema-Einstellungen, 447
Namespace-Einstellungen in der Schema Design-Ansicht, 438
Namespaces definieren, 26
Navigation in der Design-Ansicht, 50
neues erstellen, 23
Oracle Extensions, 445
Oracle Schema-Einstellungen, 446
siehe auch Schema, 428
Tamino Extensions, 448
Tamino Schema-Einstellungen, 449
Tutorial, 22
Verwaltung und Zuweisung in IBM DB2-Datenbanken, 495
von XML-Dokument generieren, 429
während der Bearbeitung des XML-Dokuments ändern, 74
XML-Datei erzeugen, 435
zu XML-Dokument zuweisen, 428
XML-Schema-Datentypen,
bei der Generierung von XML-Schema anhand einer ADO DB, 1008
bei der Generierung von XML-Schema anhand einer MS Access DB, 1003
bei der Generierung von XML-Schema anhand einer MS SQL Server DB, 1004
bei der Generierung von XML-Schema anhand einer MySQL DB, 1005
bei der Generierung von XML-Schema anhand einer ODBC DB, 1007
bei der Generierung von XML-Schema anhand einer Oracle DB, 1006
bei der Generierung von XML-Schema anhand einer Sybase DB, 1009
Konvertierung in MS Access-Datentypen, 1011
Konvertierung in MS SQL Server-Datentypen, 1013
Konvertierung in MySQL-Datentypen, 1015
Konvertierung in Oracle-Datentypen, 1017
XMLSpy, 355, 607
allgemeine Einführung, 7
Eclipse-Ansichten und Perspektiven, 250
integration, 941
plug-in registration, 647
und Eclipse, 236
und Eclipse SOAP Debugger, 247
und Eclipse XSLT Debugger, 245
Unterschiede zu Eclipse, 244
XMLSpy API,
accessing, 967
documentation, 662
overview, 663
release notes, 611
XMLSpy Enterprise Edition,
Benutzerhandbuch, 3
XMLSpy integration,
example of, 942, 943, 944
XMLSPY plug-in, 646
XMLSpyCommand,
in XMLSpyControl, 968
XMLSpyCommands,
in XMLSpyControl, 969
XMLSpyControl,
documentation of, 941
example of integration at application level, 942, 943, 944
examples of integration at document level, 946
in XMLSpyControl, 970

© 2007 Altova GmbH

XMLSpyControl,
integration at application level, 942
integration at document level, 945, 946
integration using C#, 946, 947, 948, 949
integration using HTML, 949, 950, 951, 953
object reference, 968
XMLSpyControlDocument,
in XMLSpyControl, 976
XMLSpyControlPlaceHolder,
in XMLSpyControl, 982
XMLSpyDocumentEditor,
MarkUpView, 888
XMLSpyLib, 662
Application, 684
AuthenticDataTransfer, 700
AuthenticRange, 701
AuthenticSelection (obsolete), 866
AuthenticView, 726
CodeGeneratorDlg, 740
DatabaseConnection, 745
Dialogs, 749
DocEditEvent (obsolete), 850
DocEditView (obsolete), 869
Document, 752
Documents, 777
DTDSchemaGeneratorDlg, 780
ElementList, 784
ElementListItem, 785
ExportSettings, 786
FileSelectionDlg, 788
GenerateSampleXMLDlg, 800
GridView, 804
ProjectItem, 820
SchemaDocumentationDlg, 807
SpyProject, 819
SpyProjectItems, 823
TextImportExportSettings, 824
XMLData, 841
XML-Tabellen in der Authentic-Ansicht,
Editiersymbole, 207
Verwendung, 203
XML-Text, 591
kopieren, 370
XML-Vorlagendatei,
in der Authentic-Ansicht, 84
XPath 1.0,
in XPath Evaluator, 421
XPath 2.0,
in XPath Evaluator, 421
XPath 2.0 Functions,
implementation information, 998
XPath 2.0-Funktionen,
allgemeine Informationen, 998
spezifische Funktionen siehe unter fn:, 998
XPath Evaluator,
Verwendung, 421
XPath kopieren, 373
XPath-Ausdruck-Auswertung, 112
XPath-Funktionen - Unterstützung,
für einzelne Funktionen siehe unter fn:, 999
XPaths,
für tracepoints definieren, 274
XPath-Watch-Fenster,
in XSLT/XQuery Debugger, 266
XQuery,
Ausführung, 125
Datei öffnen, 120
DB-Unterstützung, 126
Dokumentenvalidierung, 125
Eingabehilfen, 121
Funktionen für IBM DB2, 126
in Textansicht bearbeiten, 120
intelligente Editierfeatures, 124
Prozessor-Informationen, 989
Syntaxfärbung, 122
Variablen an XQuery-Dokument übergeben, 455
zur Abfrage von XML-Datenbanken, 126
XQuery 1.0 Functions,
implementation information, 998
XQuery 1.0-Funktionen,
allgemeine Informationen, 998
spezifische Funktionen siehe unter fn:, 998
XQuery 1.0-Prozessor,
Informationen, 995
XQuery Debugger,
siehe XSLT/XQuery Debugger, 252
XQuery-Ausführung, 459
XQuery-Dateien,
Dateierweiterungen in XMLSpy festlegen, 120
XQuery-Dokumente,
Ausführungsdauer analysieren, 279, 459
XQuery-Prozessor,
in Altova-Produkten, 1022
xs:QName,
siehe auch QName, 999
xsi:type,

© 2007 Altova GmbH

xsi:type,
Verwendung, 58
XSL,
siehe XSLT, 460
XSL/Query Menü, 452
XSL:FO,
Transformation, 454
xsl:preserve-space, 990
xsl:strip-space, 990
XSLT, 545
in XMLSpy ändern, 78
Prozessor, 599
Prozessor-Informationen, 989
XSLT 1.0-Prozessor,
Einschränkungen und implementierungsspezifisches Verhalten, 990
XSLT 2.0 Stylesheet,
Namespace-Deklarationen, 992
XSLT 2.0-Funktionen,
implementierungsspezifisches Verhalten, 994
spezifische Funktionen siehe unter fn:, 994
XSLT 2.0-Prozessor,
allgemeine Informationen, 992
Informationen, 992
XSLT Debugger,
aktuelle Execution Nodes einblenden, 462
aussteigen, 462
Breakpoint einfügen/löschen, 462
Breakpoints Dialogfeld, 463
Breakpoints Ein/Aus, 463
Debug-Fenster, 464
Debugger erneut starten, 461
Debugger starten, 461
Debugger stoppen, 461
Debuggersitzung beenden, 461
einsteigen, 461
Einstellungen, 464
enable/disable tracepoint, 463
in Visual Studio .NET, 231
insert/remove tracepoint, 462
überspringen, 462
XSLT Stylesheet,
öffnen, 460
zu XML-Dokument zuweisen, 459, 460
XSLT Stylesheet für FO,
zu XML-Dokument zuweisen, 460
XSLT Stylesheets,
Ausführung analysieren, 279, 459
XSLT/XQuery debuggen,
verwendete Dateien, 252
XSLT/XQuery Debugger,
Beschreibung der Benutzeroberfläche, 254
Beschreibung der Funktionsweise, 254
Einstellungen, 260
Fenster "Call Stack", 266
Fenster "Info", 268
Fenster "Messages", 267
Fenster "Templates", 267
Fenster "Variablen", 265
Fenster "XPath-Watch", 266
Fenster Trace, 268
Funktionen und Verwendung, 252
Informationsfenster, 264
Kontextfenster, 265
Sitzung starten, 262
toolbar icons, 256
Tracepoints Verwendung, 274
Verwendung von Breakpoints, 271
XSLT/XQuery Profiler, 279
XSLT-Parameter,
über Schnittstelle an Stylesheet übergeben, 455
XSLT-Prozessoren,
in Altova-Produkten, 1022
XSL-Transformation,
siehe XSLT, 76
XSLT-Transformation, 453
in FO, 454
in PDF, 454
in XMLSpy, 77
Tutorial, 76
XSLT-Datei zuweisen, 77

Z

Zeichen,
Position, 545
Zeichen-Entities,
in der HTML-Ausgabe von XSLT-Transformationen, 990
Zeichennormalisierung,
in XQuery-Dokument, 995
Zeichensatz,
Kodierung, 598
Zeile,
anhängen (Dokument Editor), 470

© 2007 Altova GmbH

Zeile,
einfügen (Dokument Editor), 470
Gehe zu, 545
kopieren (Dokument Editor), 471
löschen (Dokument Editor), 471
nach oben verschieben, 471
nach unten (Dokument Editor), 471
Zeile in Grid-Ansicht,
anhängen, 419
einfügen, 419
Zeilenlänge,
Zeilenumbruch in Textansicht, 544
Zeilennummerierung,
in Textansicht, 106
Zeilenschaltung,
siehe Eingabetaste, 219
Zeilenumbruch, 587
aktivieren/dekativiere, 544
Ein/Aus, 544
ZIP-Dateien,
Pfade für Dokumente in, 224
siehe Archivansicht, 221
Zuletzt verwendetet Dateien,
Liste, 368
Zurücksetzen,
Kürzel, 577
Menübefehle, 581
Symbolleiste & Menübefehle, 575
Zuweisen,
Kürzel zu Befehl, 577
Skript zu Projekt, 566

© 2007 Altova GmbH