AltovaXML 2013

Benutzer- und Referenzhandbuch

AltovaXML 2013 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: 2012

© 2012 Altova GmbH

Inhaltsverzeichnis

1 Einführung 3

1.1 Produkt-Features ... 4
1.2 Funktionsumfang ... 5
1.3 Systemanforderungen und Installation ... 7
1.4 Zu dieser Dokumentation ... 8

2 Verwendung 10

2.1 Befehlszeile ... 11
2.1.1 XML-Validierung und Wohlgeformtheit ... 13
2.1.2 XSLT 1.0 Transformationen ... 16
2.1.3 XSLT 2.0 Transformationen ... 18
2.1.4 XQuery 1.0 Ausführungen ... 20
2.2 COM-Schnittstelle ... 22
2.2.1 Registrieren von AltovaXML als COM Serverobjekt ... 23
2.2.2 AltovaXML Objektmodell ... 24
2.2.3 Applikation ... 25
2.2.4 XMLValidator ... 26
2.2.5 XSLT1 ... 28
2.2.6 XSLT2 ... 31
2.2.7 XQuery ... 34
2.2.8 Beispiele ... 37
Visual Basic ... 37
JScript ... 38
C++ ... 39
2.3 Java-Schnittstelle ... 41
2.3.1 Java-Beispielprojekt ... 42
2.3.2 Klassen ... 45
AltovaXMLFactory ... 45
XMLValidator ... 46
XQuery ... 48
XSLT1 ... 51
XSLT2 ... 53
2.3.3 Alte Java API (nicht mehr verwendet) ... 57

Beispiel ... 58
Schnittstellen ... 59
IAltovaXMLEngine ... 60
IAltovaXMLFactory ... 61
IExecutable ... 62
IReleasable ... 62
IXMLValidator ... 63
IXQuery ... 64
IXSLT ... 66
Klassen ... 67
AltovaXMLFactory ... 67
XMLValidator ... 69
XQuery ... 72
XSLT1 ... 76
XSLT2 ... 78
2.4 .NET-Schnittstelle ... 82
2.4.1 Allgemeine Verwendung ... 84
2.4.2 Altova.AltovaXML.XMLValidator ... 86
2.4.3 Altova.AltovaXML.XSLT1 ... 89
2.4.4 Altova.AltovaXML.XSLT2 ... 94
2.4.5 Altova.AltovaXML.XQuery ... 100
2.4.6 Beispiel ... 104
2.5 Explizite Freigabe von AltovaXML COM-Server von C# und VB.NETaus ... 107
2.6 OOXML- und ZIP-Dateien ... 108

3 Prozessorinformationen 110

3.1 Altova XML Validator ... 111
3.2 XSLT 1.0-Prozessor: Implementierungsinformationen ... 112
3.3 XSLT 2.0-Prozessor: Implementierungsinformationen ... 114
3.3.1 Allgemeine Informationen ... 115
3.3.2 XSLT 2.0-Elemente und -Funktionen ... 117
3.4 XQuery 1.0-Prozessor: Implementierungsinformationen ... 118
3.5 XPath 2.0- und XQuery 1.0-Funktionen ... 121
3.5.1 Allgemeine Informationen ... 122
3.5.2 Unterstützung von Funktionen ... 124
3.6 Erweiterungen ... 127
3.6.1 Altova Erweiterungsfunktionen ... 128
Allgemeine Funktionen ... 128

4 Lizenzinformationen 134

4.1 Electronic Software Distribution ... 135

4.2 Rechte am geistigen Eigentum 136
4.3 Lizenzvereinbarung 137

Index

Kapitel 1

Einführung

1 Einführung

Die **AltovaXML 2013 Community Edition** ist ein XML-Applikationspaket, das den Altova XML Validator, den Altova XSLT 1.0-Prozessor, den Altova XSLT 2.0-Prozessor und den Altova XQuery 1.0-Prozessor enthält. Das Paket ist kostenlos erhältlich und kann in Form einer einzigen Installationsdatei von der Altova Website heruntergeladen werden. AltovaXML dient zum Validieren von XML-Dokumenten, zum Transformieren von XML-Dokumenten mit Hilfe von XSLT Stylesheets und zum Ausführen von XQuery-Dokumenten.

AltovaXML kann über die Befehlszeile, über eine COM-Schnittstelle, in Java-Programmen und in .NET-Applikationen verwendet werden. In dieser Dokumentation wird beschrieben, wie AltovaXML in all diesen Umgebungen eingesetzt werden kann. Des Weiteren enthält diese Dokumentation Informationen über die implementierungsspezifischen Aspekte der Prozessoren in diesem Paket.

Letzte Aktualisierung: 11/13/2012

1.1 Produkt-Features

Die wichtigsten Features von AltovaXML sind:

Paket

- XML Validator, XSLT-Prozessoren und XQuery-Prozessor in einer einzigen Installationsdatei.
- Installationsdatei zum Gratis-Download von der Altova Website.
- Einfache Installation ausführbarer Dateien auf Windows-Systemen.

Befehlszeile

- Verwendung der Befehlszeile für Validierung, XSLT-Transformation und XQuery-Ausführung.
- Validierung von XML-Dokumenten gemäß den DTD- und XML-Schema-Regeln des W3C.
- Transformation von XML-Dokumenten mit Hilfe von XSLT 1.0 und XSLT 2.0 Stylesheets entsprechend den jeweiligen W3C-Spezifikationen.
- Ausführung von XQuery 1.0-Dokumenten entsprechend der W3C-Spezifikation.

COM-Schnittstelle

- Kann über eine COM-Schnittstelle verwendet werden, somit auch mit Applikationen und Script-Sprachen, die COM unterstützen.
- Die COM-Schnittstelle ist für Raw- und Dispatch-Schnittstellen implementiert.
- Über die Schnittstellen-Eigenschaften steht eine breite Palette an Funktionen für XML-Validierung, XSLT-Transformation und XQuery-Ausführung zur Verfügung.
- XML, DTD, XML Schema, XSLT und XQuery Input kann in Form von Dateien oder als Textstrings in Scripts und Applikationsdaten erfolgen.

Java-Schnittstelle

- Die AltovaXML-Funktionalität steht in Java-Klassen zur Verfügung, die in Java-Programmen verwendet werden können.
- Java-Klassen bieten Funktionen zur XML-Validierung, XSLT-Transformation und XQuery-Ausführung.

.NET-Schnittstelle

- AltovaXML ist in eine DLL eingebettet, sodass .NET-User die Funktionen von AltovaXML aufrufen können.
- Bietet von Altova signierte Primary Interop Assembly.
- Bietet eine breite Palette von Funktionen zur XML-Validierung, XSLT-Transformation und XQuery-Ausführung.
- XML, DTD, XML Schema, XSLT und XQuery Input kann in Form von Dateien oder als Textstrings in Scripts und Applikationsdaten erfolgen.

© 2012 Altova GmbH

1.2 Funktionsumfang

AltovaXML bietet die unten aufgelisteten Funktionen. Die meisten dieser Funktionen sind gebräuchlich bei der Verwendung der Befehlszeile und der COM-Schnittstelle. Ein wichtiger Unterschied ist, dass Dokumente bei Verwendung der COM-Schnittstelle anhand von Textstrings über die Applikation oder den Skripting-Code konstruiert werden können (anstatt XML-, DTD-, XML Schema, XSLT oder XQuery-Dateien zu referenzieren).

XML- und XBRL-Validierung

- Validiert das vorliegende XML-Dokument und gibt den Wert "gültig" oder "ungültig" zurück.
- Die Validierung kann gegen die DTD oder das in der XML-Datei referenzierte XML-Schema oder gegen eine externe DTD oder ein XML-Schema durchgeführt werden, das über einen Befehlszeilenparameter oder die Eigenschaft einer COM-Schnittstelle bereitgestellt wird.
- Überprüft die Wohlgeformtheit des vorliegenden XML-Dokuments. Dies erfolgt separat zur Validierung.
- Validiert XBRL-Dokumente. Das XBRL-Dokument wird anhand einer XBRL-Taxonomie (die eine `.xsd`-Datei ist) entsprechend den XBRL-Regeln validiert.

XSLT-Transformationen

- Transformierung des vorliegenden XML-Dokuments anhand des mitgelieferten XSLT 1.0 - oder XSLT 2.0-Dokuments.
- Das XML-Dokument kann über eine URL als Input bereitgestellt werden. Bei Verwendung über die COM-Schnittstelle kann das XML-Dokument auch als Textstring geliefert werden.
- Das XSLT-Dokument kann über eine URL als Input bereitgestellt werden. Bei Verwendung über die COM-Schnittstelle kann das XSLT-Dokument auch als Textstring geliefert werden.
- Gibt Ausgabedokumente unter dem definierten Pfad zurück. Bei Aufruf über die COM-Schnittstelle können die Ausgabedokumente auch als String zurückgegeben werden.
- Die XSLT-Parameter können über die Befehlszeile und über die COM-Schnittstelle bereitgestellt werden.
- Spezielle Verarbeitung mit Hilfe von Altova Erweiterungsfunktionen (u.a. in der Reporting Edition Funktionen für Diagramme).

XQuery-Ausführung

- Führt das bereitgestellte XQuery 1.0-Dokument aus, optional gegen ein in einem Befehlszeilenparameter definiertes XML-Dokument oder eine COM-Schnittstelleneigenschaft.
- Ein XQuery-Dokument kann über eine URL als Input bereitgestellt werden. Bei Verwendung über die COM-Schnittstelle kann das XQuery-Dokument als Alternative auch als Textstring bereitgestellt werden.
- Ein XML-Dokument kann über eine URL als Input bereitgestellt werden. Bei Verwendung über die COM-Schnittstelle kann das XML-Dokument als Alternative auch als Textstring bereitgestellt werden.
- Gibt Ausgabedokumente unter dem definierten Pfad zurück. Bei Aufruf über die COM-Schnittstelle können die Ausgabedokumente auch als String zurückgegeben werden.
- Externe XQuery-Variablen können über die Befehlszeile und die COM-Schnittstelle

bereitgestellt werden.

- Zu den Serialisierungsoptionen gehören: Ausgabecodierung, Ausgabemethode (d.h. ob die Ausgabe als XML, XHTML, HTML oder Text erfolgen soll), Auslassen der XML-Deklaration und Einrückung.
- Spezielle Verarbeitung mit Hilfe von Altova Erweiterungsfunktionen (u.a. in der Reporting Edition Funktionen für Diagramme).

© 2012 Altova GmbH

1.3 Systemanforderungen und Installation

Systemanforderungen
AltovaXML wird auf Windows NT, Windows XP, Windows Server 2003, Windows Server 2008, Windows Vista und Windows 7 unterstützt. Um AltovaXML über eine COM-Schnittstelle verwenden zu können, benötigt der Benutzer Benutzerrechte für die COM-Schnittstelle, d.h. die Berechtigung, die Applikation zu registrieren und die jeweiligen Applikationen und/oder Scripts ausführen zu können. Die AltovaXML Reporting Edition steht sowohl für 32-Bit- als auch für 64-Bit-Systeme zur Verfügung. Die AltovaXML Community Edition steht nur für 32-Bit-Systeme zur Verfügung.

Installation
AltovaXML ist eine selbstextrahierende Datei, die auf der Altova Website zum Download zur Verfügung steht und die AltovaXML mit den nötigen Registrierungen installiert. Nachdem Sie die Installationsdatei (`AltovaXML2013.exe`) auf Ihren Rechner heruntergeladen haben, doppelklicken Sie darauf, um die Installation zu starten. AltovaXML wird unter `Programme` im Ordner `Altova/AltovaXML2013` installiert. Alle erforderlichen Registrierungen, damit AltovaXML über eine COM-Schnittstelle, als Java-Schnittstelle und in der .NET-Umgebung verwendet werden kann, werden vom Installationsprogramm vorgenommen. Dazu gehört auch die Registrierung der AltovaXML exe-Datei als COM-Serverobjekt, die Installation der Datei `AltovaXMLLib.dll` (für die Verwendung der Java-Schnittstelle) im Verzeichnis `WINDIR\system32\` das Hinzufügen der Datei `Altova.AltovaXML.dll` zur .NET Referenzbibliothek.

Beachten Sie bitte Folgendes:

- Zur Verwendung der Befehlszeile, rufen Sie die installierte exe-Datei auf (`AltovaXML.exe`). Diese Datei kann in einen anderen Ordner auf Ihrem Rechner oder im Netzwerk kopiert werden, auf den Sie Zugriff haben und über den Sie diese Datei aufrufen können.
- Sie können AltovaXML sofort über die COM-Schnittstelle verwenden, da die installierte exe-Datei `AltovaXML_COM.exe` bereits als COM-Serverobjekt registriert wurde. Wenn Sie die exe-Datei `AltovaXML_COM.exe` in einem anderen Ordner auf Ihrem Rechner oder im Netzwerk speichern, müssen Sie sie dort manuell als COM-Serverobjekt registrieren. Eine Anleitung dazu finden Sie unter, Registrieren von AltovaXML als COM-Serverobjekt.
- Um AltovaXML über eine Java-Schnittstelle verwenden zu können, muss `AltovaXML_COM.exe` als COM-Serverobjekt registriert sein und die Java-Bibliotheken müssen sich im `classpath` befinden. Die Java-Bibliotheken sind im AltovaXML-Applikationsordner im Ordner `JavaAPI` installiert. Die Registrierung als COM-Serverobjekt wird beim Installationsvorgang automatisch durchgeführt. Beachten Sie jedoch: Wenn Sie nach der Installation den Pfad zur Datei `AltovaXML_COM.exe` ändern, müssen Sie sie unter ihrem neuen Pfad anschließend manuell als COM-Serverobjekt registrieren. Nähere Informationen dazu siehe Registrieren von AltovaXML als COM Serverobjekt und Java-Schnittstelle.

1.4 Zu dieser Dokumentation

Diese Dokumentation ist die offizielle Produktdokumentation zu AltovaXML und enthält ausführliche Informationen darüber. Sie ist in die folgenden Abschnitte gegliedert:

- In der Einführung werden die Features des Produkts AltovaXML, seine Funktionen und die wichtigsten Systemanforderungen für die Verwendung von AltovaXML sowie dessen Installation beschrieben.
- Im Abschnitt Verwendung wird beschrieben, wie man AltovaXML über die Befehlszeile und eine COM-Schnittstelle verwendet. Im Abschnitt Befehlszeile finden Sie nähere Informationen über die Syntax, mit der man die verschiedenen Funktionen von AltovaXML aufrufen kann. Im Abschnitt COM-Schnittstelle finden Sie Informationen darüber, wie AltovaXML als COM-Schnittstelle verwendet werden kann; Sie finden darin eine ausführliche Beschreibung des Objektmodells, seiner Schnittstellen und der Schnittstelleneigenschaften. Der Abschnitt Java-Schnittstelle enthält eine Beschreibung zur Verwendung von AltovaXML mit Java sowie eine Auflistung der definierten Java-Schnittstellen und -Klassen. Der Abschnitt .NET-Schnittstelle enthält eine Beschreibung der Verwendung und eine Auflistung der verschiedenen Methoden und Eigenschaften, die verwendet werden können.
- Der Abschnitt Prozessor-Informationen enthält eine Beschreibung der implementierungsspezifischen Aspekte der verschiedenen Prozessoren, die die Komponenten von AltovaXML bilden. Jeder Prozessor wird separat beschrieben.

© 2012 Altova GmbH

Kapitel 2

Verwendung

2 Verwendung

Nachdem Sie AltovaXML heruntergeladen und im gewünschten Ordner installiert haben, können Sie es auf folgende Arten verwenden:

- Durch Aufruf der Anwendung über die Befehlszeile,
- Durch Verwendung der Anwendung über eine COM-Schnittstelle,
- Durch Verwendung der Anwendung über eine Java-Schnittstelle und
- Durch Verwendung der Anwendung in der .NET-Umgebung.

© 2012 Altova GmbH

2.1 Befehlszeile

Um AltovaXML über die Befehlszeile verwenden zu können, muss die exe-Datei (`AltovaXML.exe`) in einem Ordner auf Ihrem Rechner oder dem Netzwerk installiert sein, bzw. dorthin kopiert worden sein und Sie müssen Zugriff auf diesen Ordner haben. Die allgemeine Syntax zum Aufrufen der Applikation ist:

```
AltovaXML functionality arg1 ... argN [options]
```

wobei

`AltovaXML`	Ruft die Applikation auf.
`functionality`	Gibt an, ob die Funktion XML-Validierung, Wohlgeformtheitsprüfung, XSLT 1.0-Transformation, XSLT 2.0-Transformation oder XQuery 1.0-Ausführung aufgerufen wird. Die jeweiligen Werte sind `-validate` (oder `-v`), `-wellformed` (oder `-w`), `-xslt1`, `-xslt2`, `-xquery` (oder `-xq`).
`arg1 ... argN`	Die Argumente der aufgerufenen Funktionalität.
`options`	Jede Funktionalität hat ihre eigenen Optionen. Diese Optionen sind in den entsprechenden Unterabschnitten dieses Abschnitts beschrieben.

Allgemeine Optionen

`-help`, `-h`, oder `-?`	Zeigt Informationen zur Verwendung an, z.B. eine Liste aller Argumente und Optionen.
`-version`, `-ver`	Zeigt die Programmversion an.

Es stehen die folgenden Funktionalitäten zur Verfügung. Die zulässigen Argumente und Optionen für die einzelnen Funktionalitäten sind in den folgenden Abschnitten näher beschrieben:

- XML-Validierung und Wohlgeformtheit
- XSLT 1.0-Transformationen
- XSLT 2.0-Transformationen
- XQuery 1.0-Ausführung

Zusammenfassung zur Verwendung

Im Folgenden finden Sie einen Überblick über die Verwendung der Befehlszeile. Nähere Informationen finden Sie im jeweiligen Abschnitt.

Verwendung des Altova XML Validators

- `-validate <Dateiname> [-schema <Dateiname> | -dtd <Dateiname>]`
- `-wellformed <Dateiname>`

Verwendung des Altova XSLT 1.0-Prozessors

- `-xslt1 <Dateiname> -in <Dateiname> [-param name=value] [-out <Dateiname>]`

Verwendung des Altova XSLT 2.0-Prozessors

- `-xslt2 <Dateiname> -in <Dateiname> [-param name=value] [-out <Dateiname>]`

Verwendung des Altova XQuery 1.0-Prozessors

- `-xquery <Dateiname> [-in <Dateiname>] [-param name=value] [-out <Dateiname>] [serialization options]`

Hinweis: Wenn der Dateiname oder der Pfad ein Leerzeichen enthält, sollte der gesamte Pfadname in Anführungszeichen gesetzt werden, z.B.: `"c:\My Files\MyXML.xml"` oder `"c:\MyFiles\My XML.xml"`.

© 2012 Altova GmbH

2.1.1 XML-Validierung und Wohlgeformtheit

XML-Validierungssyntax
Die Syntax zum Aufrufen der **XML-Validierung** lautet:

```
AltovaXML -validate xmlfile [-schema schemafile | -dtd dtdfile]
   [options]
```

wobei

`AltovaXML`	die Applikation aufruft
`-validate` (or `-v`)	angibt, dass der Altova XML Validator zum Validieren der Datei "XML-Datei" zu verwenden ist.

Es stehen die folgenden Optionen zur Verfügung:

`-schema` (oder `-s`)	gibt an, dass die XML-Schema-Datei `Schemadatei` für die Validierung zu verwenden ist.
`-dtd` (or `-d`)	definiert, dass die DTD-Datei `DTD-Datei` für die Validierung zu verwenden ist.
`-xbrlConsistency` (oder `-xc`)	die Semantik von XBRL-Dokumenten überprüft.

Mapping-Optionen
Über den XML-Katalog und die globalen Altova-Ressourcen stehen die folgenden Mapping-Optionen zur Verfügung. (Globale Altova-Ressourcen können nur verwendet werden, wenn ein Produkt, das globale Altova Ressourcen unterstützt, wie z.B. Altova XMLSpy, installiert ist).

`-catalog` (oder `-c`) `[<filename>]`	Aktiviert die Katalogzuordnung unter Verwendung des angegebenen Katalogs. Wenn keine Datei definiert ist, wird ein Katalog namens `RootCatalog.xml` aus dem AltovaXML-Applikationsordner als Standardordner verwendet.
`-globalresources` (oder `-gr`) `[<filename>]`	Aktiviert die Zuordnung von globalen Altova-Ressourcen unter Verwendung der angegebenen globalen Ressourcen-XML-Datei oder - falls keine Datei definiert wird - unter Verwendung von `GlobalResources.xml` in `My Documents/Altova`.
`-globalresourceconfig` (oder `-gc`) `[<name>]`	Definiert die aktive Konfigurationsdatei für globale Ressourcen.

Anmerkung zu globalen Ressourcen
Zum Auswählen einer Ressource unter Verwendung der Methode "globale Altova-Ressourcen" sind zwei Einstellungen erforderlich:

- Die XML-Datei für globale Ressourcen enthält Definitionen der globalen Ressourcen. Diese Datei kann mit der Option `-globalresources` (oder `-gr`) definiert werden. Falls keine Datei angegeben wird, so wird die Datei `GlobalResources.xml` im Ordner `My Documents/Altova` verwendet.
- Jede globale Ressource in der XML-Datei für globale Ressourcen kann mehrere Konfigurationen haben, wobei jede Konfiguration einer Ressource zugeordnet ist. Mit Hilfe der Option `-globalresourceconfig` (oder `-gc`) können Sie angeben, welche

Konfiguration verwendet werden soll. Mit der Erweiterung wird definiert, welche Ressource verwendet werden soll.

Hinweis:

- Wenn kein XML-Schema bzw. keine DTD-Datei als Befehlszeilenoption definiert ist, muss im XML-Dokument selbst ein XML-Schema oder eine DTD-Datei definiert sein.
- Wenn in der Befehlszeile ein XML-Schema oder eine DTD-Datei angegeben sind **und** in der XML-Datei ein XML-Schema oder eine DTD-Datei referenziert ist, dann wird die in der Befehlszeilenoption angegebene Datei für die Validierung verwendet.
- Wenn ein XBRL-Instanzdokument validiert wird, wird dies anhand der XBRL-Taxonomie, bei der es sich um eine .xsd-Datei handelt, durchgeführt. Wenn zusätzlich zur Syntaxvalidierung eine semantische Validierung erforderlich ist, verwenden Sie die Option `-xbrlConsistency`.

Syntax für die Wohlgeformtheitsprüfung
Die Syntax zum Aufrufen der **Wohlgeformtheitsprüfung** lautet:

`AltovaXML -wellformed` ***`XML-Datei`***

wobei

`AltovaXML`	die Applikation aufruft
`-wellformed` (oder `-w`)	angibt, dass zur Überprüfung der Wohlgeformtheit der Datei `XML-Datei` der Altova XML Validator zu verwenden ist.

Mapping-Optionen
Über den XML-Katalog und die globalen Altova-Ressourcen stehen die folgenden Mapping-Optionen zur Verfügung. (Globale Altova-Ressourcen können nur verwendet werden, wenn ein Produkt, das globale Altova Ressourcen unterstützt, wie z.B. Altova XMLSpy, installiert ist).

`-catalog` (oder `-c`) `[<filename>]`	Aktiviert die Katalogzuordnung unter Verwendung des angegebenen Katalogs. Wenn keine Datei definiert ist, wird ein Katalog namens `RootCatalog.xml` aus dem AltovaXML-Applikationsordner als Standardordner verwendet.
`-globalresources` (oder `-gr`) `[<filename>]`	Aktiviert die Zuordnung von globalen Altova-Ressourcen unter Verwendung der angegebenen globalen Ressourcen-XML-Datei oder - falls keine Datei definiert wird - unter Verwendung von `GlobalResources.xml` in `My Documents/Altova`.
`-globalresourceconfig` (oder `-gc`) `[<name>]`	Definiert die aktive Konfigurationsdatei für globale Ressourcen.

Anmerkung zu globalen Ressourcen
Zum Auswählen einer Ressource unter Verwendung der Methode "globale Altova-Ressourcen" sind zwei Einstellungen erforderlich:

- Die XML-Datei für globale Ressourcen enthält Definitionen der globalen Ressourcen. Diese Datei kann mit der Option `-globalresources` (oder `-gr`) definiert werden. Falls keine Datei angegeben wird, so wird die Datei `GlobalResources.xml` im Ordner `My Documents/Altova` verwendet.

 © 2012 Altova GmbH

- Jede globale Ressource in der XML-Datei für globale Ressourcen kann mehrere Konfigurationen haben, wobei jede Konfiguration einer Ressource zugeordnet ist. Mit Hilfe der Option `-globalresourceconfig` (oder `-gc`) können Sie angeben, welche Konfiguration verwendet werden soll. Mit der Erweiterung wird definiert, welche Ressource verwendet werden soll.

Beispiele

- `AltovaXML -validate test.xml -schema testschema.xsd`
- `AltovaXML -v test.xml -dtd testdtd.dtd`
- `AltovaXML -wellformed test.xml`
- `AltovaXML -w test.xml`
- `AltovaXML -v test.xml -dtd testdtd.dtd -c MyCatalog.xml`
- `AltovaXML -validate test.xml -schema testschema.xsd -xc`

Hinweis: Bei Verwendung von AltovaXML in Batch-Befehlen, beachten Sie bitte Folgendes:

- Der Rückgabecode des letzten ausgeführten Befehls wird in der Variable `errorlevel` gespeichert, deren Wert mit einem Batch-Befehl wie z.B. `ECHO %errorlevel%` abgerufen werden kann.
- Die Rückgabecodes sind `0` = wohlgeformt/gültig; `1` = nicht wohlgeformt/ungültig.

2.1.2 XSLT 1.0 Transformationen

Syntax
Die Syntax zum Aufrufen der **XSLT 1.0 Transformation** lautet:

```
AltovaXML -xslt1 xslt-Datei -in xml-Datei [-out Ausgabedatei]
   [options]
```

wobei

`AltovaXML`	die Applikation aufruft.
`-xslt1`	angibt, dass für eine XSLT-Transformation der Altova XSLT 1.0 Prozessor zu verwenden ist; der Prozessor verwendet die XSLT 1.0-Datei *`xslt-Datei`* für die Transformation.
`-in`	angibt, dass die XML-Datei `xml-Datei` transformiert werden soll und deren Pfad angibt.
`-out`	diei `Ausgabedatei` und deren Pfad angibt. Wenn diese Option weggelassen wird, wird die Ausgabe in die Standardausgabe geschrieben.

Es stehen die folgenden Optionen zur Verfügung:

`-param`	Ruft die Anweisung `paramname=XPath expression` auf. Der `-param` Switch wird vor den einzelnen globalen Parametern verwendet. Wenn in einem XPath-Ausdruck - egal ob in einem XPath-Ausdruck selbst oder in einem String Literal im Ausdruck - Leerzeichen enthalten sind, müssen doppelte Anführungszeichen verwendet werden. Siehe Beispiele.
`-xslstack`	Die Stack-Größe ist die maximale Tiefe, bis zu der Anweisungen ausgeführt werden. Sie kann über den Wert `-xslstack` geändert werden. Der zulässige Mindestwert beträgt 100. Die Standard-Stack-Größe ist 1000. Bei Überschreitung der Stack-Größe während einer Transformation wird ein Fehler ausgegeben.
`-namedTemplate` (oder `-n`)	Definiert die named Template am Anfang. Das Argument wird durch ein Leerzeichen von seinem Wert getrennt. Beispiel: `-namedTemplate MyTemplate`
`-mode` (oder `-m`)	Definiert den Vorlagenmodus am Anfang. Sets the initial template mode.Das Argument wird durch ein Leerzeichen von seinem Wert getrennt. Beispiel: `-mode MyMode`

Mapping-Optionen
Über den XML-Katalog und die globalen Altova-Ressourcen stehen die folgenden Mapping-Optionen zur Verfügung. (Globale Altova-Ressourcen können nur verwendet werden, wenn ein Produkt, das globale Altova Ressourcen unterstützt, wie z.B. Altova XMLSpy, installiert ist).

`-catalog` (oder `-c`) `[<filename>]`	Aktiviert die Katalogzuordnung unter Verwendung des angegebenen Katalogs. Wenn keine Datei definiert ist, wird ein Katalog namens `RootCatalog.xml` aus dem AltovaXML-Applikationsordner als Standardordner verwendet.
`-globalresources` (oder `-gr`) `[<filename>]`	Aktiviert die Zuordnung von globalen Altova-Ressourcen unter Verwendung der angegebenen globalen Ressourcen-XML-Datei oder - falls keine Datei definiert wird - unter Verwendung von

 © 2012 Altova GmbH

	GlobalResources.xml in My Documents/Altova.
-globalresourceconfig (oder -gc) [<name>]	Definiert die aktive Konfigurationsdatei für globale Ressourcen.

Anmerkung zu globalen Ressourcen
Zum Auswählen einer Ressource unter Verwendung der Methode "globale Altova-Ressourcen" sind zwei Einstellungen erforderlich:

- Die XML-Datei für globale Ressourcen enthält Definitionen der globalen Ressourcen. Diese Datei kann mit der Option -globalresources (oder -gr) definiert werden. Falls keine Datei angegeben wird, so wird die Datei GlobalResources.xml im Ordner My Documents/Altova verwendet.
- Jede globale Ressource in der XML-Datei für globale Ressourcen kann mehrere Konfigurationen haben, wobei jede Konfiguration einer Ressource zugeordnet ist. Mit Hilfe der Option -globalresourceconfig (oder -gc) können Sie angeben, welche Konfiguration verwendet werden soll. Mit der Erweiterung wird definiert, welche Ressource verwendet werden soll.

Hinweis:

- Die XSLT-Datei muss in der Befehlszeilenanweisung definiert werden; eine XSLT-Datei, die in einer <?xml-stylesheet?> Verarbeitungsanweisung im XML-Dokument referenziert wird, wird nicht automatisch verwendet.
- Wenn der -out Parameter nicht angegeben wird, wird die Ausgabe in die Standardausgabe geschrieben.

Beispiele

- `AltovaXML -xslt1 test.xslt -in test.xml -out testout.xml`
- `AltovaXML -xslt1 test.xslt -in test.xml -out testout.xml -c MyCatalog.xml`
- `AltovaXML -xslt1 test.xslt -in test.xml -out testout.xml -param date=//node/@att1`
- `AltovaXML -xslt1 test.xslt -in test.xml -out testout.xml -param date="//node/@att1 | //node/@att2"`
- `AltovaXML -xslt1 test.xslt -in test.xml -out testout.xml -param date=node/@att1 -param title='stringwithoutspace'`
- `AltovaXML -xslt1 test.xslt -in test.xml -out testout.xml -param date=node/@att1 -param title="'string with spaces'"`

2.1.3 XSLT 2.0 Transformationen

Syntax
Die Syntax zum Aufrufen der **XSLT 2.0 Transformation** lautet:

```
AltovaXML -xslt2 xslt-Datei -in xml-Datei [-out Ausgabedatei]
   [options]
```

wobei

`AltovaXML`	die Applikation aufruft.
`-xslt2`	angibt, dass für eine XSLT-Transformation der Altova XSLT 2.0 Prozessor zu verwenden ist; der Prozessor verwendet die XSLT 2.0-Datei *`xslt-Datei`* für die Transformation.
`-in`	angibt, dass die XML-Datei `xml-Datei` transformiert werden soll und deren Pfad angibt.
`-out`	die `Ausgabedatei` und deren Pfad angibt. Wenn diese Option weggelassen wird, wird die Ausgabe in die Standardausgabe geschrieben.

Es stehen die folgenden Optionen zur Verfügung:

`-param`	Ruft die Anweisung `paramname=XPath expression` auf. Der `-param` Switch wird vor den einzelnen globalen Parametern verwendet. Wenn in einem XPath-Ausdruck - egal ob in einem XPath-Ausdruck selbst oder in einem String Literal im Ausdruck - Leerzeichen enthalten sind, müssen doppelte Anführungszeichen verwendet werden. Siehe Beispiele
`-xslstack`	Die Stack-Größe ist die maximale Tiefe, bis zu der Anweisungen ausgeführt werden. Sie kann über den Wert `-xslstack` geändert werden. Der zulässige Mindestwert beträgt 100. Die Standard-Stack-Größe ist 1000. Bei Überschreitung der Stack-Größe während einer Transformation wird ein Fehler ausgegeben.
`-namedTemplate` (oder `-n`)	Definiert die named Template am Anfang. Das Argument wird durch ein Leerzeichen von seinem Wert getrennt. Beispiel: `-namedTemplate MyTemplate`
`-mode` (oder `-m`)	Definiert den Vorlagenmodus am Anfang. Sets the initial template mode.Das Argument wird durch ein Leerzeichen von seinem Wert getrennt. Beispiel: `-mode MyMode`

Mapping-Optionen
Über den XML-Katalog und die globalen Altova-Ressourcen stehen die folgenden Mapping-Optionen zur Verfügung. (Globale Altova-Ressourcen können nur verwendet werden, wenn ein Produkt, das globale Altova Ressourcen unterstützt, wie z.B. Altova XMLSpy, installiert ist).

`-catalog` (oder `-c`) `[<filename>]`	Aktiviert die Katalogzuordnung unter Verwendung des angegebenen Katalogs. Wenn keine Datei definiert ist, wird ein Katalog namens `RootCatalog.xml` aus dem AltovaXML-Applikationsordner als Standardordner verwendet.
`-globalresources` (oder `-gr`) `[<filename>]`	Aktiviert die Zuordnung von globalen Altova-Ressourcen unter Verwendung der angegebenen globalen Ressourcen-XML-Datei oder - falls keine Datei definiert wird - unter Verwendung von

 © 2012 Altova GmbH

	GlobalResources.xml in My Documents/Altova.
-globalresourceconfig (oder -gc) [<name>]	Definiert die aktive Konfigurationsdatei für globale Ressourcen.

Anmerkung zu globalen Ressourcen
Zum Auswählen einer Ressource unter Verwendung der Methode "globale Altova-Ressourcen" sind zwei Einstellungen erforderlich:

- Die XML-Datei für globale Ressourcen enthält Definitionen der globalen Ressourcen. Diese Datei kann mit der Option -globalresources (oder -gr) definiert werden. Falls keine Datei angegeben wird, so wird die Datei GlobalResources.xml im Ordner My Documents/Altova verwendet.
- Jede globale Ressource in der XML-Datei für globale Ressourcen kann mehrere Konfigurationen haben, wobei jede Konfiguration einer Ressource zugeordnet ist. Mit Hilfe der Option -globalresourceconfig (oder -gc) können Sie angeben, welche Konfiguration verwendet werden soll. Mit der Erweiterung wird definiert, welche Ressource verwendet werden soll.

Hinweis:

- Die XSLT-Datei muss in der Befehlszeilenanweisung definiert werden; eine XSLT-Datei, die in einer <?xml-stylesheet?> Verarbeitungsanweisung im XML-Dokument referenziert wird, wird nicht automatisch verwendet.
- Wenn der -out Parameter nicht angegeben wird, wird die Ausgabe in die Standardausgabe geschrieben.
- Der XSLT 2.0-Prozessor kann zur Verarbeitung eines XSLT 1.0 Stylesheet im Rückwärtskompatibilitätsmodus verwendet werden. Die Ausgabe kann sich allerdings von der eines XSLT 1.0-Prozessors, der dasselbe XSLT 1.0 Stylesheet verarbeitet, unterscheiden.

Beispiele

- `AltovaXML -xslt2 test.xslt -in test.xml -out testout.xml`
- `AltovaXML -xslt2 test.xslt -in test.xml -out testout.xml -c MyCatalog.xml`
- `AltovaXML -xslt2 test.xslt -in test.xml -out testout.xml -param date=//node/@att1`
- `AltovaXML -xslt2 test.xslt -in test.xml -out testout.xml -param date="//node/@att1 | //node/@att2"`
- `AltovaXML -xslt2 test.xslt -in test.xml -out testout.xml -param date=node/@att1 -param title='stringwithoutspace'`
- `AltovaXML -xslt2 test.xslt -in test.xml -out testout.xml -param date=node/@att1 -param title="'string with spaces'"`

2.1.4 XQuery 1.0 Ausführungen

Syntax
Die Syntax zum Aufrufen einer XQuery 1.0 Ausführung lautet:

```
AltovaXML -xquery xquery-Datei [-in XML-Input-Datei -out
   Ausgabedatei] [options]
```

wobei

`AltovaXML`	die Applikation aufruft.
`-xquery` (oder `-xq`)	angibt, dass für eine XQuery-Ausführung der Datei `xquery-Datei` der Altova XQuery 1.0-Prozessor zu verwenden ist.
`-in`	die XML-Input-Datei definiert.
`-out`	die Ausgabedatei und deren Pfad definiert. Wenn diese Option weggelassen wird, wird die Ausgabe in die Standardausgabe geschrieben.

Es stehen die folgenden Optionen zur Verfügung:

`-var`	Gibt eine externe Variable und deren Wert an. Hat die Form `name=value`. Es können beliebig viele externe Variablen verwendet werden, jeder muss jedoch das Schlüsselwort `-var` vorangestellt werden. Variablenwerte müssen Strings sein, die der lexikalischen Form des Datentyps entsprechen, als der die Variable deklariert wurde.
`-xparam`	Gibt einen XQuery-Parameternamen und den Wert des Parameters an. Hat die Form `name=XPathExpression`. Setzen Sie den XPath-Ausdruck in doppelte Anführungszeichen, wenn der Ausdruck Leerzeichen enthält. Setzen Sie String-Literalzeichen im XPath-Ausdruck in einfache Anführungszeichen. Es können beliebig viele Parameter verwendet werden, jedoch muss jedem das Schlüsselwort `-xparam` vorangestellt werden.
`-outputMethod` (oder `-om`)	Serialisierungsoption zum Definieren des Ausgabeart. Gültige Werte sind `xml`, `html`, `xhtml` und `text`. Standardwert ist `xml`.
`-omitXMLDeclaration` (or `-od`)	Serialisierungsoption, um festzulegen, ob die XML-Deklaration in der Ausgabe enthalten sein soll oder nicht. Gültige Werte sind `yes` und `no`. Standardwert ist `yes`.
`-outputIndent` (or `-oi`)	Serialisierungsoption, um festzulegen, ob die Ausgabe Einrückungen enthalten soll oder nicht. Gültige Werte sind `yes` und `no`. Standardwert ist `no`.
`-outputEncoding` (or `-oe`)	Serialisierungsoption zur Definition des Zeichensatzes der Ausgabe. Gültige Werte sind Namen in der IANA Zeichensatz-Registrierdatei. Standardwert ist `UTF-8`.

Mapping-Optionen
Über den XML-Katalog und die globalen Altova-Ressourcen stehen die folgenden Mapping-Optionen zur Verfügung. (Globale Altova-Ressourcen können nur verwendet werden, wenn ein Produkt, das globale Altova Ressourcen unterstützt, wie z.B. Altova XMLSpy, installiert ist).

`-catalog` (oder `-c`)	Aktiviert die Katalogzuordnung unter Verwendung des angegebenen

 © 2012 Altova GmbH

[<filename>]	Katalogs. Wenn keine Datei definiert ist, wird ein Katalog namens RootCatalog.xml aus dem AltovaXML-Applikationsordner als Standardordner verwendet.
-globalresources (oder -gr) [<filename>]	Aktiviert die Zuordnung von globalen Altova-Ressourcen unter Verwendung der angegebenen globalen Ressourcen-XML-Datei oder - falls keine Datei definiert wird - unter Verwendung von GlobalResources.xml in My Documents/Altova.
-globalresourceconfig (oder -gc) [<name>]	Definiert die aktive Konfigurationsdatei für globale Ressourcen.

Anmerkung zu globalen Ressourcen
Zum Auswählen einer Ressource unter Verwendung der Methode "globale Altova-Ressourcen" sind zwei Einstellungen erforderlich:

- Die XML-Datei für globale Ressourcen enthält Definitionen der globalen Ressourcen. Diese Datei kann mit der Option -globalresources (oder -gr) definiert werden. Falls keine Datei angegeben wird, so wird die Datei GlobalResources.xml im Ordner My Documents/Altova verwendet.
- Jede globale Ressource in der XML-Datei für globale Ressourcen kann mehrere Konfigurationen haben, wobei jede Konfiguration einer Ressource zugeordnet ist. Mit Hilfe der Option -globalresourceconfig (oder -gc) können Sie angeben, welche Konfiguration verwendet werden soll. Mit der Erweiterung wird definiert, welche Ressource verwendet werden soll.

Hinweis: Wenn der -out Parameter nicht angegeben wird, wird die Ausgabe in die Standardausgabe geschrieben.

Beispiele

- `AltovaXML -xquery testquery.xq -out testout.xml`
- `AltovaXML -xquery testquery.xq -in products.xml -out testout.xml -var company=Altova -var date=2006-01-01`
  - `AltovaXML -xquery testquery.xq -out testout.xml -xparam source=" doc( 'c:\test\books.xml' )//book "`
- `AltovaXML -xquery testquery.xq -in products.xml -out testout.xml -var company=Altova -omitXMLDeclaration no -oe ASCII`

2.2 COM-Schnittstelle

Wenn AltovaXML als COM Serverobjekt registriert ist, kann das Programm von Applikationen und Script-Sprachen aus aufgerufen werden, die Programmierunterstützung für COM-Aufrufe bieten. Dies ist nützlich, da auf diese Art die XML-Dokumentvalidierung, XSLT-Transformationen (XSLT 1.0 und XSLT 2.0) und XQuery 1.0 Dokumentausführungen von AltovaXML über eine breite Palette von Benutzerapplikationen ausgeführt werden können.

Um AltovaXML mit Applikationen und Script-Sprachen verwenden zu können, die eine COM-Schnittstelle haben, müssen Sie AltovaXML zuerst als COM Serverobjekt registrieren. Nähere Informationen dazu finden Sie unter Registrieren von AltovaXML als COM Serverobjekt.

Das AltovaXML Objektmodell und seine Eigenschaften werden in den folgenden Unterabschnitten dieses Abschnitts beschrieben. (Beachten Sie: Sie können sowohl die Raw-Schnittstelle als auch die Dispatch-Schnittstelle von COM verwenden. Die Raw-Schnittstelle wird für Programmiersprachen wie z.B. C++ verwendet. Die Dispatch-Schnittstelle wird für Script-Sprachen (wie z.B. JavaScript) verwendet, die das Übergeben von Parametern über eine Referenz nicht gestatten.) Sie können AltovaXML daher mit folgenden Sprachen verwenden:

- Skriptsprachen wie z.B. JavaScript oder anderen Skriptsprachen, die die COM-Schnittstelle unterstützen.
- Programmiersprachen wie C++ oder andere, die die COM-Schnittstelle unterstützen.
- Java und .NET, für die Schnittstellen als Wrapper erstellt werden, wobei Klassen rund um die COM-Schnittstelle erstellt werden.

Dieser Abschnitt über die Verwendung der COM-Schnittstelle endet mit einer Gruppe von Beispielen, wie verschiedene Funktionen von AltovaXML über verschiedene Benutzerapplikationen aufgerufen werden können.

Beispiele
Zusätzliche Beispieldateien finden Sie im Ordner `Examples` des Applikationsordners.

 © 2012 Altova GmbH

2.2.1 Registrieren von AltovaXML als COM Serverobjekt

Wenn Sie AltovaXML 2013 registrieren, wird `AltovaXML_COM.exe` automatisch als COM-Serverobjekt registriert. Wenn Sie den Pfad von `AltovaXML_COM.exe` ändern müssen, sollten Sie AltovaXML am besten deinstallieren und dann im gewünschten Ordner neu installieren. Auf diese Art wird die notwendige Deregistrierung und Registrierung vom Installer durchgeführt. Wenn Sie `AltovaXML_COM.exe` auf einen anderen Rechner kopieren, müssen Sie AltovaXML unter dem neuen Pfad manuell als COM-Serverobjekt registrieren, wie im Folgenden erklärt. Es wird in dieser Beschreibung davon ausgegangen, dass AltovaXML erfolgreich installiert wurde.

Manuelle Registrierung
So registrieren Sie AltovaXML als COM-Serverobjekt:

1. Kopieren Sie `AltovaXML_COM.exe` in den gewünschten Ordner. Wenn sich der Ordner nicht auf Ihrem lokalen Rechner befindet, mappen Sie ihn auf einen Netzwerkordner.
2. Öffnen Sie die Windows Eingabeaufforderung oder klicken Sie im Menü "Start" auf **Ausführen...**.
3. Registrieren Sie die Applikation mit Hilfe des Parameters `/regserver` als COM-Serverobjekt. Wenn `AltovaXML_COM.exe` sich z.B. im Ordner `c:\AltovaXML` befindet, geben Sie ein:

   ```
   c:\AltovaXML\AltovaXML_COM.exe /regserver
   ```

   und drücken Sie die Eingabetaste.

Überprüfen, ob die Registrierung erfolgreich war
Wenn die Registrierung erfolgreich war, sollte die Registrierdatei die Klassen `AltovaXML.Application` und `AltovaXML.Application.1` enthalten. Diese beiden Klassen finden Sie normalerweise unter `HKEY_LOCAL_MACHINE\SOFTWARE\Classes`.

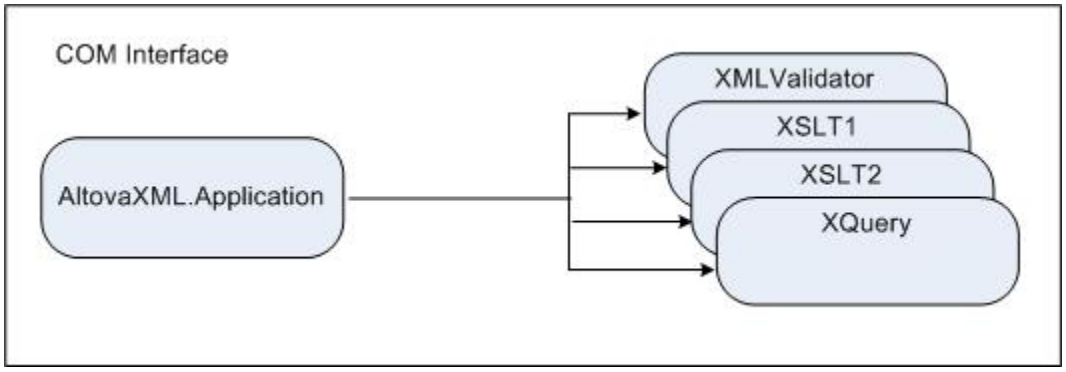
Registrierung manuell löschen
Wenn Sie `AltovaXML_COM.exe` manuell registriert haben und diese Registrierung nun löschen möchten, müssen Sie dies manuell tun. Um die Registrierung von AltovaXML manuell zu löschen, rufen Sie die Applikation über den Parameter `/unregserver` auf. Wenn sich die AltovaXML exe-Datei z.B. im Ordner `c:\AltovaXML` befindet, öffnen Sie die Windows-Eingabeaufforderung und geben Sie ein `c:\AltovaXML\AltovaXML_COM.exe /unregserver` und drücken Sie die Eingabetaste. Sie können im Registrierungs-Editor überprüfen, ob die entsprechenden Klassen gelöscht wurden.

Anmerkung: Wenn AltovaXML vom Installer registriert wurde, sollte die Registrierung vom Installer gelöscht werden, d.h. durch Deinstallation von AltovaXML vom Rechner.

2.2.2 AltovaXML Objektmodell

Um die Funktionalitäten von AltovaXML verwenden zu können, benötigen Sie die Applikationsschnittstelle. Dieses Objekt enthält die vier Objekte, die die AltovaXML-Funktionalitäten bereitstellen: XML-Validierung, XSLT 1.0-Transformationen, XSLT 2.0-Transformationen und XQuery 1.0-Dokumentverarbeitung. Diese Objekte haben zwei Schnittstellen: die Dispatch-Schnittstelle und die Raw-Schnittstelle, sodass sie sowohl in Skriptsprachen als auch in Applikationen verwendet werden können.

Das Objektmodell der AltovaXML API wird im folgenden Diagramm dargestellt.



Im Folgenden sehen Sie die Hierarchie des Objektmodells. Die fünf Schnittstellen werden in den jeweiligen Abschnitten näher beschrieben. Die Eigenschaften und deren Verwendung werden im Abschnitt zu der entsprechenden Schnittstelle behandelt.

- `Applikation`
  - `XMLValidator`
  - `XSLT1`
  - `XSLT2`
  - `XQuery`

Hinweis:
Beachten Sie die folgenden allgemeinen Hinweise zur Verwendung der COM-Schnittstelle:

- Der Begriff XML-Dokument bezieht sich nicht nur auf ein in einer XML-Datei enthaltenes XML-Dokument, sondern auch auf ein mit der Eigenschaft `InputXMLFromText` erstelltes XML-Dokument.
- Eigenschaften, die als Input einen Ressourcenpfad nehmen, akzeptieren absolute Pfade sowie das HTTP- und FTP-Protokoll.
- Bei Verwendung relativer Pfade durch eine Methode, um eine Ressource zu finden, sollte die Auflösung des relativen Pfads im aufrufenden Modul definiert sein.

© 2012 Altova GmbH

2.2.3 Applikation

Beschreibung
`AltovaXML.Application` ist die Root für alle anderen Objekte. Sie ist das einzige Objekt, das Sie mit der `CreateObject` Funktion (von VisualBasic) oder anderen ähnlichen COM-Funktionen erstellen können.

Eigenschaften
`AltovaXML.Application` hat die vier unten aufgelisteten Eigenschaften. Jede dieser Funktionen gibt die Schnittstelle für die jeweilige Komponente zurück. Die Details zu den einzelnen Schnittstellen finden Sie in den entsprechenden Abschnitten weiter unten.

- XMLValidator
- XSLT1
- XSLT2
- XQuery

Methoden
Mit den folgenden Methoden, die im application-Objekt zur Verfügung stehen, können Kataloge für Dokument-Lookup-Zwecke hinzugefügt werden. Nachdem die Kataloge hinzugefügt wurden, werden sie so lange zu Lookup Zwecken herangezogen, bis der COM Server beendet wird. Hinzugefügte Kataloge können nicht mehr entfernt werden.

`app.AddXMLCatalogDefault()`
Fügt den Altova-Standardkatalog `RootCatalog.xml` zu den Katalogen hinzu.

`app.AddXMLCatalogFromFile( string catalogfilename )`
Fügt den durch `catalogfilename` identifizierten Katalog zu den Katalogen hinzu.

`app.AddXMLCatalogFromText( string catalogtext )`
Fügt den Katalog mit dem Inhalt `catalogtext` zu den Katalogen hinzu.

Beispiele
Im Folgenden sehen Sie ein Visual Basic-Script, das zuerst das AltovaXML-Objekt erstellt und anschließend die Eigenschaften der Applikationsschnittstelle aufruft.

```
Sub CommandButton1_Click()
Set objAltovaXML = CreateObject("AltovaXML.Application")

    objAltovaXML.XMLValidator.InputXMLFileName =
"c:\AltovaXML\test.xml"
    Sheet1.Cells(5, 2) = objAltovaXML.XMLValidator.IsValid

    objAltovaXML.XSLT1.InputXMLFromText = "<?xml version='1.0'
encoding='UTF-8'?><a><b/></a>"
    objAltovaXML.XSLT1.XSLFileName = "c:\workarea\altova_xml\1.xslt"
    Sheet1.Cells(6, 2) =
objAltovaXML.XSLT1.ExecuteAndGetResultAsString

End Sub
```

2.2.4 XMLValidator

Beschreibung
Die `XMLValidator` Schnittstelle stellt Methoden bereit, um Folgendes zu testen:

- die Wohlgeformtheit eines XML-Dokuments.
- die Gültigkeit eines XML-Dokuments entsprechend einer DTD oder eines XML-Schemas, die/das von innerhalb des XML-Dokuments referenziert wird.
- die Gültigkeit eines XML-Dokuments entsprechend einer DTD oder eines XML-Schema, die/das extern über den Code bereitgestellt wird.
- die Gültigkeit eines XBRL-Dokuments entsprechend einer XBRL-Taxonomie (einer `.xsd`-Datei)

All diese Methoden geben den Booleschen Wert `TRUE` oder `FALSE` zurück. *Siehe Beispiele unten*.

Hinweis: Wo String-Inputs als URLs interpretiert werden sollen, sollten absolute Pfade verwendet werden. Bei Verwendung eines relativen Pfads sollte im aufrufenden Modul ein Mechanismus definiert werden, der den relativen Pfad auflöst.

Methoden
Es stehen die folgenden Methoden zur Verfügung:

`IsWellFormed) as Boolean`
`IsWellFormed` überprüft die Wohlgeformtheit des XML-Dokuments. Gibt `TRUE` zurück, wenn das XML-Dokument wohlgeformt ist und `FALSE,` wenn es nicht wohlgeformt ist.

`IsValid) as Boolean`
`IsValid` validiert das XML-Dokument gegen die DTD oder das XML-Schema, die/das im XML-Dokument referenziert wird. Gibt `TRUE` zurück, wenn das XML-Dokument gültig ist, `FALSE`, wenn es ungültig ist. Um das Dokument gegen eine DTD oder ein XML-Schema zu validieren, das/die im XML-Dokument nicht referenziert wird, verwenden Sie die Methode `IsValidWithExternalSchemaOrDTD`.

`IsValidWithExternalSchemaOrDTD) as Boolean`
`IsValidWithExternalSchemaOrDTD` validiert das XML-Dokument gegen die DTD oder das XML-Schema, die/das durch eine der folgenden Eigenschaften bereitgestellt wurde: `SchemaFileName`, `DTDFileName`, `SchemaFromText` oder `DTDFromText`. Wurden für mehrere dieser Eigenschaften Werte definiert, so verwendet die Methode `IsValidWithExternalSchemaOrDTD` die als letzte definierte Eigenschaft. Wenn das XML-Dokument gültig ist, wird `TRUE` zurückgegeben, wenn es ungültig ist, `FALSE`. Um das Dokument gegen eine DTD oder ein XML-Schema zu validieren, die/das im XML-Dokument referenziert wird, verwenden Sie die Methode `IsValid`.

Hinweis: Die Validierung und Wohlgeformtheitsprüfung muss immer erfolgen, nachdem das XML-Dokument und/oder die DTD oder das XML-Schema-Dokument den jeweiligen Eigenschaften zugewiesen wurde.

Eigenschaften
Es sind die folgenden Eigenschaften definiert:

`InputXMLFileName`
Ein String-Input, der als URL zum Auffinden der XML-Datei gelesen wird, die validiert werden soll.

 © 2012 Altova GmbH

SchemaFileName
Ein String-Input, der als URL zum Auffinden der XML-Schemadatei gelesen wird, gegen die das XML-Dokument validiert werden soll.

DTDFileName
Ein String-Input, der als URL zum Auffinden der DTD-Datei gelesen wird, gegen die das XML-Dokument validiert werden soll.

InputXMLFromText
Ein String-Input, der ein XML-Dokument erstellt.

SchemaFromText
Ein String-Input, der ein XML-Schema-Dokument erstellt.

DTDFromText
Ein String-Input, der ein DTD-Dokument erstellt.

LastErrorMessage
Gibt die letzte Fehlermeldung zurück.

TreatXBRLInconsistenciesAsErrors
Gibt semantische XBRL-Inkonsistenzen als Fehler zurück, wenn diese Eigenschaft auf "True" gesetzt wird. Die Standardeinstellung ist `False`.

Beispiele
Im Folgenden sehen Sie eine Visual Basic-Prozedur, die zeigt, wie Methoden und Eigenschaften der `XMLValidator` Schnittstelle verwendet werden können. Dieser Code ist zur Verwendung in einem Makro in einem MS Excel-Arbeitsblatt gedacht und referenziert Zellen im Arbeitsblatt, die den Pfad zu Eingabe- oder Ausgabedaten angeben. Die Datei `c:\AltovaXML\test.xml` enthält in diesem Fall eine Referenz auf eine DTD.

```
Sub CommandButton1_Click()
Set objAltovaXML = CreateObject("AltovaXML.Application")

    objAltovaXML.XMLValidator.InputXMLFromText = "<?xml version='1.0'
encoding='UTF-8'?><a><b/></a>"
    Sheet1.Cells(4, 2) = objAltovaXML.XMLValidator.IsWellFormed

    objAltovaXML.XMLValidator.InputXMLFileName =
"c:\AltovaXML\test.xml"
    Sheet1.Cells(5, 2) = objAltovaXML.XMLValidator.IsValid

    objAltovaXML.XMLValidator.InputXMLFileName =
"c:\AltovaXML\test.xml"
    objAltovaXML.XMLValidator.DTDFileName = "c:\AltovaXML\test.dtd"
    Sheet1.Cells(6, 2) =
objAltovaXML.XMLValidator.IsValidWithExternalSchemaOrDTD

    objAltovaXML.XMLValidator.InputXMLFromText = "<?xml version='1.0'
encoding='UTF-8'?><a><b/></a>"
    objAltovaXML.XMLValidator.DTDFileName = "c:\AltovaXML\test.dtd"
    Sheet1.Cells(7, 2) =
objAltovaXML.XMLValidator.IsValidWithExternalSchemaOrDTD
End Sub
```

2.2.5 XSLT1

Beschreibung
Die `XSLT1`-Schnittstelle stellt Methoden und Eigenschaften zum Ausführen einer XSLT 1.0-Transformation mittels des Altova XSLT 1.0-Prozessors bereit. Das Ergebnis kann in einer Datei oder als String zurückgegeben werden. Über die Schnittstelle können außerdem XSLT-Parameter an das XSLT-Stylesheet übergeben werden. Die URLs von XML- und XSLT-Dateien können als Strings über Schnittstelleneigenschaften bereitgestellt werden. Alternativ dazu können XML- und XSLT-Dokumente innerhalb des Script- oder Programmiercodes als Textstrings konstruiert werden. *Siehe Beispiele unten*.

Hinweis: Wo String-Inputs als URLs interpretiert werden sollen, sollten absolute Pfade verwendet werden. Bei Verwendung eines relativen Pfads sollte im aufrufenden Modul ein Mechanismus definiert werden, der den relativen Pfad auflöst.

Methoden
Es stehen die folgenden Methoden zur Verfügung:

`Execute(OutputFileName as String)`
`void execute(String outputFilename)`
`Execute` führt die XSLT 1.0-Transformation aus und speichert das Ergebnis als Ausgabedatei, deren Name und Pfad als Input-String für die `Execute` Methode bereitgestellt wird. z.B.:
`Execute("C:\OutputDoc.xml").`

`ExecuteAndGetResultAsString() as String`
`ExecuteAndGetResultAsString` führt eine XSLT 1.0 Transformation aus und gibt das Ergebnis als UTF-16 Textstring zurück.

`AddExternalParameter(ParamName as String, ParamValue as String)`
Nimmt einen Parameternamen und den Wert dieses Parameters als Input-Argumente. Die einzelnen externen Parameter und deren Werte müssen in einem separaten Aufruf der Methode definiert werden. Wenn derselbe Parametername in mehrere Aufrufen definiert ist, wird der im letzten Aufruf definierte Wert verwendet. Da Parameterwerte XPath-Ausdrücke sind, müssen Parameterwerte, die Strings sind, in einfache Anführungszeichen eingeschlossen werden. In diesem Beispiel werden zwei Parameter verwendet:

```
AddExternalParameter("Param1","'http://www.altova.com/'");
AddExternalParameter("Param2","concat('http://www.altova.com/',
MyFile/@url)");
```

Siehe auch die Beispiele unten.

`ClearExternalParameterList()`
Es sollte kein Argument bereitgestellt werden. Die Methode `ClearExternalParameterList` löscht die Liste der externen Parameter, die mit `AddExternalParameter` Methoden erstellt wurden.

Hinweis: Die Transformation muss immer erfolgen, nachdem das XML-Dokument und das XSLT-Dokument zugewiesen wurden.

Eigenschaften
Es sind die folgenden Eigenschaften definiert:

`InputXMLFileName`
Ein String-Input, der als URL gelesen wird, über die die zu transformierende XML-Datei gefunden wird.

 © 2012 Altova GmbH

XSLFileName
Ein String-Input, der als URL zum Auffinden der XSLT-Datei gelesen wird, die für die Transformation verwendet werden soll.

InputXMLFromText
Ein String-Input, der ein XML-Dokument erstellt.

XSLFromText
Ein String-Input, der ein XSLT-Dokument erstellt.

XSLStackSize
Die Stack-Größe ist die maximale Tiefe, bis zu der Anweisungen ausgeführt werden sollen. Die Stack-Größe kann über die Eigenschaft `XSLStackSize` geändert werden. Die zulässige Mindest-Stack-Größe ist 100. Die Standard-Stack-Größe ist 1000. Bei Überschreitung der Stack-Größe während einer Transformation wird ein Fehler ausgegeben.

LastErrorMessage
Gibt die letzte Fehlermeldung zurück.

JavaExtensionsEnabled
Aktiviert die Java-Erweiterungen. Sie können festlegen, ob Java-Erweiterungen aktiviert werden sollen oder nicht, indem Sie den Wert `true` oder `false` (keine Berücksichtigung der Groß- und Kleinschreibung) als Argument Variant_Bool liefern.

DotNetExtensionsEnabled
Aktiviert die .NET-Erweiterungen. Sie können festlegen, ob .NET-Erweiterungen aktiviert werden sollen oder nicht, indem Sie den Wert `true` oder `false` (keine Berücksichtigung der Groß- und Kleinschreibung) als Argument Variant_Bool liefern.

Beispiele
Im Folgenden sehen Sie eine Visual Basic-Prozedur, die zeigt, wie die verschiedenen Methoden und Eigenschaften oder `XSLT1`-Schnittstellen verwendet werden können. Dieser Code dient zur Verwendung als Makro in einem MS Excel-Arbeitsblatt und referenziert Zellen, im Arbeitsblatt, die Pfade zu Input- und Output-Daten definieren.

```
Sub CommandButton1_Click()
Set objAltovaXML = CreateObject("AltovaXML.Application")

    objAltovaXML.XSLT1.InputXMLFromText = "<?xml version='1.0'
encoding='UTF-8'?>
        <a><b/></a>"
    objAltovaXML.XSLT1.XSLFileName = "c:\AltovaXML\test.xslt"
    objAltovaXML.XSLT1.Execute "c:\AltovaXML\test_result.xml

    objAltovaXML.XSLT1.XSLStackSize = "500"
    objAltovaXML.XSLT1.InputXMLFromText = "<?xml version='1.0'
encoding='UTF-8'?>
        <company><name/><year>2005</year></company>"
    objAltovaXML.XSLT1.XSLFileName = "c:\AltovaXML\test.xslt"
    objAltovaXML.XSLT1.AddExternalParameter "web", "'www.altova.com'"
    objAltovaXML.XSLT1.AddExternalParameter "year", "/company/year"
    Sheet1.Cells(6, 2) =
objAltovaXML.XSLT1.ExecuteAndGetResultAsString
    objAltovaXML.XSLT1.ClearExternalParameterList
    objAltovaXML.XSLT1.AddExternalParameter "web",
"'www.nanonull.com'"
    objAltovaXML.XSLT1.AddExternalParameter "year", "/company/year"
```

```
    Sheet1.Cells(7, 2) =
objAltovaXML.XSLT1.ExecuteAndGetResultAsString
End Sub
```


© 2012 Altova GmbH

2.2.6 XSLT2

Beschreibung
Die `XSLT2`-Schnittstelle stellt Methoden und Eigenschaften zum Ausführen einer XSLT 2.0-Transformation mittels des Altova XSLT 2.0-Prozessors bereit. Das Ergebnis kann in einer Datei oder als String zurückgegeben werden. Über die Schnittstelle können außerdem XSLT-Parameter an das XSLT-Stylesheet übergeben werden. Die URLs von XML- und XSLT-Dateien können als Strings über Schnittstelleneigenschaften bereitgestellt werden. Alternativ dazu können XML- und XSLT-Dokumente innerhalb des Script- oder Programmiercodes als Textstrings konstruiert werden. *Siehe Beispiele unten*.

Hinweis:
- Wo String-Inputs als URLs interpretiert werden sollen, sollten absolute Pfade verwendet werden. Bei Verwendung eines relativen Pfads sollte im aufrufenden Modul ein Mechanismus definiert werden, der den relativen Pfad auflöst.
- Der XSLT 2.0-Prozessor kann zur Verarbeitung eines XSLT 1.0 Stylesheet im Rückwärtskompatibilitätsmodus verwendet werden. Die Ausgabe kann sich allerdings von der eines XSLT 1.0-Prozessors, der dasselbe XSLT 1.0 Stylesheet verarbeitet, unterscheiden.

Methoden
Es stehen die folgenden Methoden zur Verfügung:

`Execute(OutputFileName as String)`
`void` **`execute`**`(String outputFilename)`
`Execute` führt die XSLT 2.0-Transformation aus und speichert das Ergebnis als Ausgabedatei, deren Name und Pfad als Input-String für die `Execute` Methode bereitgestellt werden. z.B.:
`Execute("C:\OutputDoc.xml").`

`ExecuteAndGetResultAsString() as String`
`ExecuteAndGetResultAsString` führt eine XSLT 2.0 Transformation aus und gibt das Ergebnis als UTF-16 Textstring zurück.

`AddExternalParameter(ParamName as String, ParamValue as String)`
Nimmt einen Parameternamen und den Wert dieses Parameters als Input-Argumente. Die einzelnen externen Parameter und deren Werte müssen in einem separaten Aufruf der Methode definiert werden. Wenn derselbe Parametername in mehrere Aufrufen definiert ist, wird der im letzten Aufruf definierte Wert verwendet. Da Parameterwerte XPath-Ausdrücke sind, müssen Parameterwerte, die Strings sind, in einfache Anführungszeichen eingeschlossen werden. *Siehe Beispiel unten*. Beachten Sie in den Beispielen, dass der Parameter `date` einen Wert erhält, der eine XPath 2.0-Funktion ist (`current-date()`). In diesem Beispiel werden zwei Parameter verwendet:

```
AddExternalParameter("Param1","'http://www.altova.com/'");
AddExternalParameter("Param2","concat('http://www.altova.com/',
MyFile/@url)");
```

Siehe auch die Beispiele unten.

`ClearExternalParameterList()`
Es sollte kein Argument bereitgestellt werden. Die Methode `ClearExternalParameterList` löscht die Liste der externen Parameter, die mit `AddExternalParameter` Methoden erstellt wurden.

`InitialTemplateName`
Definiert die anfangs verwendete named Template. Das Argument ist der Name der Vorlage,

an der die Verarbeitung beginnen soll. Z.B.: `InitialNamedTemplate ="MyNamedTemplate".`

`InitialTemplateMode`
Definiert den anfangs verwendeten Modus für die Verarbeitung. Das Argument ist der Name des erforderlichen Anfangsmodus. Vorlagen mit diesem Moduswert werden verarbeitet. z.B. `InitialTemplateMode="MyMode".`

Hinweis: Die Transformation muss immer erfolgen, nachdem das XML-Dokument und das XSLT-Dokument zugewiesen wurden.

Eigenschaften
Es sind die folgenden Eigenschaften definiert:

`InputXMLFileName`
Ein String-Input, der als URL gelesen wird, über die die zu transformierende XML-Datei gefunden wird.

`XSLFileName`
Ein String-Input, der als URL zum Auffinden der XSLT-Datei gelesen wird, die für die Transformation verwendet werden soll.

`InputXMLFromText`
Ein String-Input, der ein XML-Dokument erstellt.

`XSLFromText`
Ein String-Input, der ein XSLT-Dokument erstellt.

`XSLStackSize`
Die Stack-Größe ist die maximale Tiefe, bis zu der Anweisungen ausgeführt werden sollen. Die Stack-Größe kann über die Eigenschaft `XSLStackSize` geändert werden. Die zulässige Mindest-Stack-Größe ist 100. Die Standard-Stack-Größe ist 1000. Bei Überschreitung der Stack-Größe während einer Transformation wird ein Fehler ausgegeben.

`LastErrorMessage`
Gibt die letzte Fehlermeldung zurück.

`JavaExtensionsEnabled`
Aktiviert die Java-Erweiterungen. Sie können festlegen, ob Java-Erweiterungen aktiviert werden sollen oder nicht, indem Sie den Wert `true` oder `false` (keine Berücksichtigung der Groß- und Kleinschreibung) als Argument Variant_Bool liefern.

`DotNetExtensionsEnabled`
Aktiviert die .NET-Erweiterungen. Sie können festlegen, ob .NET-Erweiterungen aktiviert werden sollen oder nicht, indem Sie den Wert `true` oder `false` (keine Berücksichtigung der Groß- und Kleinschreibung) als Argument Variant_Bool liefern

Beispiele
Im Folgenden sehen Sie eine Visual Basic-Prozedur, die zeigt, wie die verschiedenen Methoden und Eigenschaften oder `XSLT2`-Schnittstellen verwendet werden können. Dieser Code dient zur Verwendung als Makro in einem MS Excel-Arbeitsblatt und referenziert Zellen, im Arbeitsblatt, die Pfade zu Input- und Output-Daten definieren.

```
Sub CommandButton1_Click()
Set objAltovaXML = CreateObject("AltovaXML.Application")

    objAltovaXML.XSLT2.InputXMLFromText = "<?xml version='1.0'
```

 © 2012 Altova GmbH

```
encoding='UTF-8'?>
        <a><b/></a>"
    objAltovaXML.XSLT2.XSLFileName = "c:\AltovaXML\test.xslt"
    Sheet1.Cells(7, 2) =
objAltovaXML.XSLT2.ExecuteAndGetResultAsString

    objAltovaXML.XSLT2.XSLStackSize = "500"
    objAltovaXML.XSLT2.InputXMLFromText = "<?xml version='1.0'
encoding='UTF-8'?>
        <company><name/><year>2005</year></company>"
    objAltovaXML.XSLT2.XSLFileName = "c:\workarea\AltovaXML\2.xslt"
    objAltovaXML.XSLT2.AddExternalParameter "date", "current-date()"
    objAltovaXML.XSLT2.AddExternalParameter "hq", "'Vienna, Austria'"
    Sheet1.Cells(8, 2) =
objAltovaXML.XSLT2.ExecuteAndGetResultAsString
    objAltovaXML.XSLT2.AddExternalParameter "web",
"'www.nanonull.com'"
    objAltovaXML.XSLT2.AddExternalParameter "year", "/company/year"
    objAltovaXML.XSLT2.Execute
"c:\workarea\AltovaXML\test_result_xslt2.xml"
    Sheet1.Cells(9, 2) =
objAltovaXML.XSLT2.ExecuteAndGetResultAsString
End Sub
```

2.2.7 XQuery

Beschreibung
Die `XQuery`-Schnittstelle stellt Methoden und Eigenschaften zum Ausführen einer XQuery 1.0-Transformation mittels des Altova XQuery-Prozessors bereit. Das Ergebnis kann in einer Datei oder als String zurückgegeben werden. Über die Schnittstelle können außerdem XQuery-Variablen an das XQuery-Dokument übergeben werden. Die URLs von XQuery- und XML-Dateien können als Strings über Schnittstelleneigenschaften bereitgestellt werden. Alternativ dazu können XML- und XQuery-Dokumente innerhalb des Script- oder Programmiercodes als Textstrings konstruiert werden. *Siehe Beispiele unten*.

Hinweis: Wo String-Inputs als URLs interpretiert werden sollen, sollten absolute Pfade verwendet werden. Bei Verwendung eines relativen Pfads sollte im aufrufenden Modul ein Mechanismus definiert werden, der den relativen Pfad auflöst.

Methoden
Es stehen die folgenden Methoden zur Verfügung:

`Execute(OutputFileName as String)`
`void execute(String outputFilename)`
`Execute` führt die XSLT 2.0-Transformation aus und speichert das Ergebnis als Ausgabedatei, deren Name und Pfad als Input-String für die `Execute` Methode bereitgestellt werden. z.B.: `Execute("C:\OutputDoc.xml").`

`ExecuteAndGetResultAsString() as String`
`ExecuteAndGetResultAsString` führt eine XQuery 1.0-Transformation aus und gibt das Ergebnis als UTF-16 Textstring zurück.

`AddExternalVariable(VarName as String, VarValue as String)`
Nimmt einen Variablennamen und den Wert dieser Variable als Input-Argumente. Die einzelnen externen Variablen und deren Werte müssen in einem separaten Aufruf der Methode definiert werden. Variablen müssen im XQuery-Dokument deklariert werden, optional mit einer Typdeklaration. Unabhängig von der Typdeklaration für die externe Variable im XQuery-Dokument sind für den Variablenwert, der für die Methode `AddExternalVariable` bereitgestellt wird, keine speziellen Trennzeichen wie z.B. Anführungszeichen erforderlich (*siehe Beispiel unten).* Die lexikalische Form muss jedoch der des erwarteten Typs entsprechen (so muss z.B. eine Variable vom Typ `xs:date` einen Wert in der lexikalischen Form `2004-01-31` haben; ein Wert in der lexikalischen Form `2004/Jan/01` verursacht dagegen einen Fehler). Beachten Sie, dass dies auch bedeutet, dass Sie eine XQuery 1.0-Funktion (z.B. `current-date()`) nicht als den Wert einer externen Variable verwenden können (da die lexikalische Form der Funktion in der eigenen Schreibweise entweder nicht dem erforderlichen Datentyp entspricht - wenn der Datentyp in der Deklaration der externen Variable definiert ist - oder als String gelesen wird, wenn der Datentyp nicht definiert ist. Wenn eine externe Variable bereitgestellt wird, die den Namen einer existierenden (nicht gelöschten) Variable hat, so wird ein Fehler ausgegeben. Wenn derselbe Variablenname in mehrere Aufrufen definiert ist, wird der im letzten Aufruf definierte Wert verwendet.

`AddExternalVariableAsXPath(VarName as String, VarValue as String)`
Nimmt einen Variablennamen und den Wert dieser Variable als Input-Argumente. Ist ähnlich wie die Methode `AddExternalVariable`,mit der Ausnahme, dass `AddExternalVariableAsXPath` als XPath 2.0 Ausdruck ausgewertet wird. Auf diese Art können auch Nodes und Sequenzen mit mehr als einem Element verarbeitet werden.

`ClearExternalVariableList`
Es sollte kein Argument bereitgestellt werden. Die Methode `ClearExternalVariableList` löscht die Liste der externen Variablen, die mit `AddExternalVariable` Methoden erstellt wurden.

 © 2012 Altova GmbH

Hinweis: Die Definition des optionalen XML-Dokuments muss immer vor der XQuery-Ausführung erfolgen.

Eigenschaften

Es sind die folgenden Eigenschaften definiert:

XQueryFileName
Ein String-Input, der als URL zum Auffinden der auszuführenden XQuery-Datei gelesen wird. Wenn sowohl die `XQueryFileName` Eigenschaft als auch die `QueryFromText` Eigenschaft definiert ist, so wird diejenige Eigenschaft verwendet, die in der Codesequenz als letzte definiert ist.

InputXMLFileName
Ein String-Input, der als URL zum Auffinden der in die Query zu ladenden XML-Datei gelesen wird. XQuery Navigationsausdrücke werden in Beziehung auf den Dokumentnode dieses XML-Dokuments ausgewertet. Wenn sowohl die `InputFileName` Eigenschaft als auch die die `InputXMLFromText` Eigenschaft definiert ist, so wird diejenige Eigenschaft verwendet, die in der Codesequenz als letzte definiert ist.

XQueryFromText
Ein String-Input, der ein XQuery-Dokument erstellt. Wenn sowohl die `XQueryFileName` Eigenschaft als auch die die `XQueryFromText` Eigenschaft definiert ist, so wird diejenige Eigenschaft verwendet, die in der Codesequenz als letzte definiert ist.

InputXMLFromText
Ein String-Input, der ein XML-Dokument erstellt. XQuery Navigationsausdrücke werden in Bezug auf den Dokument-Node dieses XML-Dokuments ausgewertet. Wenn sowohl die `InputFileName` Eigenschaft als auch die `InputXMLFromText` Eigenschaft definiert ist, so wird diejenige Eigenschaft verwendet, die in der Codesequenz als letzte definiert ist.

LastErrorMessage
Gibt die letzte Fehlermeldung zurück.

JavaExtensionsEnabled
Aktiviert die Java-Erweiterungen. Sie können festlegen, ob Java-Erweiterungen aktiviert werden sollen oder nicht, indem Sie den Wert `true` oder `false` (keine Berücksichtigung der Groß- und Kleinschreibung) als Argument Variant_Bool liefern.

DotNetExtensionsEnabled
Aktiviert die .NET-Erweiterungen. Sie können festlegen, ob .NET-Erweiterungen aktiviert werden sollen oder nicht, indem Sie den Wert `true` oder `false` (keine Berücksichtigung der Groß- und Kleinschreibung) als Argument Variant_Bool liefern

Hinweis: Wenn ein XML-Dokument definiert ist und nicht für eine neue XQuery-Ausführung erforderlich ist, sollte dies durch Zuweisung eines leeren Strings gelöscht werden.

Es sind die folgenden Serialisierungsoptionen definiert:

OutputMethod
Die benötigte Ausgabemethode kann durch Bereitstellung des erforderlichen Werts als String-Argument definiert werden. Gültige Werte sind: `xml`, `xhtml`, `html` und `text`. Z.B.: `objAltovaXML.XQuery.OutputMethod = "xml"`. Wenn der Wert ungültig ist, wird er ignoriert. Die Standardausgabemethode ist `xml`.

OutputOmitXMLDeclaration
Sie können angeben, ob die XML-Deklaration in der Ausgabe inkludiert werden soll oder nicht. Geben Sie dazu `true` oder `false` (Groß- und Kleinschreibung ist irrelevant) als Boolesches Argument an. Z.B: `objAltovaXML.XQuery.OutputOmitXMLDeclaration = "FALSE"`. Wenn der Wert ungültig ist, wird ein Fehler ausgegeben. Die Standardoption ist `TRUE`.

OutputIndent
Sie können festlegen, ob die Ausgabe Einrückungen enthalten soll oder nicht. Geben Sie dazu `true` oder `false` (Groß- und Kleinschreibung ist irrelevant) als Boolesches Argument an. z.B.: `objAltovaXML.XQuery.OutputIndent = "TRUE"`. Wenn der Wert ungültig ist, wird ein Fehler ausgegeben. Die Standardoption ist False.

OutputEncoding
Die erforderliche Ausgabecodierung kann durch Angabe des Codierungswerts als String-Argument definiert werden. Z.B.: `objAltovaXML.XQuery.OutputEncoding = "UTF-8"`. Wenn der Wert ungültig ist, wird er ignoriert. Die Standardausgabecodierung ist UTF-8.

Hinweis: Bei den Serialisierungsoptionen gibt es Unterschiede bei der Verwendung der Raw-Schnittstelle und der Dispatch-Schnittstelle. Wenn bei Verwendung der Raw-Schnittstelle kein Argument mit diesen Eigenschaften bereitgestellt wird, so wird der aktuelle Wert der Eigenschaft zurückgegeben. Die Eingabe müsste in etwa so aussehen: `put_OutputOption( VARIANT_BOOL bVal )` bzw. so `VARIANT_BOOL bVal = get_OutputOption()`, um Werte zu setzen und abzurufen. Bei der Dispatch-Schnittstelle können Sie `b = myXQuery.OutputOption` verwenden, um Werte abzurufen und `myXQuery.OutputOption = b` um Werte zu definieren. Bei der Dispatch-Schnittstelle würde z.B. `Sheet1.Cells(10, 2) = objAltovaXML.XQuery.OutputEncoding` die aktuelle Ausgabecodierung abrufen.

Beispiele
Im Folgenden sehen Sie eine einzelne Visual Basic-Prozedur, die zeigt, wie die verschiedenen Methoden und Eigenschaften der `XQuery`-Schnittstelle verwendet werden können. Dieser Code war als Verwendung als Makro in einem MS Excel-Arbeitsblatt gedacht und die Referenzen auf Zellen im Arbeitsblatt geben den Pfad zu Input- oder Output-Daten an.

```
Sub CommandButton1_Click()
Set objAltovaXML = CreateObject("AltovaXML.Application")

    objAltovaXML.XQuery.InputXMLFileName = "c:\AltovaXML\test.xml"
    objAltovaXML.XQuery.XQueryFromText = " xquery version '1.0';
       declare variable $string as xs:string external;
       declare variable $num as xs:decimal external;
       declare variable $date as xs:date external;
       $string, ' ', 2*$num, ' ', $date "
    objAltovaXML.XQuery.AddExternalVariable "string", "A string"
    objAltovaXML.XQuery.AddExternalVariable "num", "2.1"
    objAltovaXML.XQuery.AddExternalVariable "date", "2005-04-21"
    Sheet1.Cells(10, 2) = objAltovaXML.XQuery.OutputEncoding
    objAltovaXML.XQuery.OutputMethod = "text"
    Sheet1.Cells(11, 2) = objAltovaXML.XQuery.OutputMethod
    objAltovaXML.XQuery.OutputIndent = "TRUE"
    Sheet1.Cells(12, 2) = objAltovaXML.XQuery.OutputIndent
    objAltovaXML.XQuery.OutputOmitXMLDeclaration = "FALSE"
    Sheet1.Cells(13, 2) = objAltovaXML.XQuery.OutputOmitXMLDeclaration
    Sheet1.Cells(14, 2) =
objAltovaXML.XQuery.ExecuteAndGetResultAsString
End Sub
```

 © 2012 Altova GmbH

2.2.8 Beispiele

Dieser Abschnitt enthält Beispielcode in (i) Visual Basic für ein Excel Makro; (ii) JScript; und (iii) C++. In diesen Beispielen sehen Sie, wie AltovaXML mit einer COM-Schnittstelle verwendet werden kann.

Nähere Beispiele finden Sie in den Beispieldateien im Applikationsordner im Ordner `Examples`.

Visual Basic

Das folgende Visual Basic-Beispiel ist der Code für ein Makro in einem Excel-Arbeitsblatt (*Screenshot unten*). Das Makro wurde der Schaltfläche `Run Expressions` zugewiesen. Wenn Sie auf die Schaltfläche klicken, wird der Visual Basic-Code ausgeführt.

	A	B
1	XQuery or XML in Application	Result
2	element a {for $i in (-3 to 3) return -$i}	<a>3 2 1 0 -1 -2 -3
3	<node>6;154;738-34</node>	6.154.738 34
4		A code-generated string
5	Run Expressions	

Codebeispiel
Der folgende Visual Basic-Code verwendet die `XQuery`-Schnittstelle.

```
Sub CommandButton1_Click()
Set objAltovaXML = CreateObject("AltovaXML.Application")

    objAltovaXML.XQuery.XQueryFromText = Sheet1.Cells(2, 1)
    Sheet1.Cells(2, 2) =
objAltovaXML.XQuery.ExecuteAndGetResultAsString

    objAltovaXML.XQuery.InputXMLFromText = Sheet1.Cells(3, 1)
    objAltovaXML.XQuery.XQueryFromText = "translate(node, ';-', '. ')"
    Sheet1.Cells(3, 2) =
objAltovaXML.XQuery.ExecuteAndGetResultAsString

    objAltovaXML.XQuery.InputXMLFromText = "<a myAttr='A
code-generated string'/>"
    objAltovaXML.XQuery.XQueryFromText = "string(/a/@*)"
    Sheet1.Cells(4, 2) =
objAltovaXML.XQuery.ExecuteAndGetResultAsString
End Sub
```

Wenn Sie im Excel-Arbeitsblatt auf die Schaltfläche **Run Expressions** klicken, werden die folgenden drei XQuery-Anweisungen ausgeführt:

1. Der Input für die `XQueryFromText` Eigenschaft ist ein XQuery-Ausdruck, der als Text aus dem Excel-Arbeitsblatt Zelle 2A geholt wird. Die `ExecuteAndGetResultAsString` Eigenschaft führt den XQuery-Ausdruck aus und platziert das Ergebnis in das Excel-Arbeitsblatt, Zelle 2B.
2. Der Input für die `InputXMLFromText` Eigenschaft ist ein XML-Fragment aus dem Excel-Arbeitsblatt, Zelle 3A. Der XQuery-Ausdruck wird an die `XQueryFromText` Eigenschaft direkt im Code übergeben. Das Ergebnis wird in das Excel-Arbeitsblatt, Zelle 3 B platziert.

3. Die `InputXMLFromText` Eigenschaft erstellt eine XML-Baumstruktur anhand des bereitgestellten XML-Fragments. Der XQuery-Ausdruck wird an die `XQueryFromText` Eigenschaft direkt im Code übergeben und das Ergebnis wird in das Excel-Arbeitsblatt, Zelle 4B platziert.

JScript

Im Folgenden sehen Sie eine JScript-Codedatei, die zeigt, wie AltovaXML über die COM-Schnittstelle verwendet werden kann.

Codebeispiel

```
// ////////////// global variables /////////////////
var objAltovaXML = null;

// ////////////////////// Helpers ///////////////////////////////

function Exit(strErrorText)
{
   WScript.Echo(strErrorText);

   if (objAltovaXML != null)
      objAltovaXML.Quit();

   WScript.Quit(-1);
}

function ERROR(strText, objErr)
{
   if (objErr != null)
      Exit ("ERROR: (" + (objErr.number & 0xffff) + ")" + objErr.description +
" - " + strText);
   else
      Exit ("ERROR: " + strText);
}

function CreateGlobalObjects ()
{
   // create the AltovaXML connection
   // if there is a running instance of AltovaXML (that never had a
connection) - use it
   // otherwise, we automatically create a new instance
   try
   {
      objAltovaXML = WScript.GetObject("", "AltovaXML.Application");
      //WScript.Echo("Successfully accessing AltovaXML.Application");
   }
   catch(err)
   {
   WScript.Echo(err)
      { Exit("Can't access or create AltovaXML.Application"); }
   }
}


// ////////////////////// MAIN ///////////////////////////////

CreateGlobalObjects();

objAltovaXML.XQuery.InputXMLFromText = " \
  <bib> \
  <book year=\"1994\">  \
     <title>TCP/IP Illustrated</title>  \
     <author><last>Stevens</last><first>W.</first></author>  \
```

```
     <publisher>AW</publisher>  \
     <price>65.95</price>  \
  </book>  \
  <book year=\"1992\">  \
     <title>Advanced Programming in the Unix Environment</title>  \
     <author><last>Stevens</last><first>W.</first></author>  \
     <publisher>AW</publisher>  \
     <price>65.95</price> \
  </book>  \
  <book year=\"2000\">  \
     <title>Data on the Web</title>  \
     <author><last>Abiteboul</last><first>Serge</first></author>  \
     <author><last>Abiteboul</last><first>Serge</first></author>  \
     <author><last>Abiteboul</last><first>Serge</first></author>  \
     <publisher>John Jameson Publishers</publisher>  \
     <price>39.95</price>  \
  </book>  \
  <book year=\"1999\">  \
     <title>Digital TV</title>  \

<editor><last>Gassy</last><first>Viktor</first><affiliation>CITI</affiliation>
</editor>  \
     <publisher>Kingston Academic Press</publisher>  \
     <price>129.95</price>  \
  </book>  \
 </bib> ";


objAltovaXML.XQuery.XQueryFromText = "\
 (: Filename: xmpQ1.xq :)  \
 (: Source: http://www.w3.org/TR/xquery-use-cases/#xmp-data :)  \
 (: Section: 1.1.1.9 Q1 :)  \
 (: List books published by AW after 1991, including their year and title.:)
\
 <bib>  \
 { \
     for $b in /bib/book  where $b/publisher = \"AW\" and $b/@year > 1991   \
              return <book year=\"{ $b/@year }\">     { $b/title }    </book>
\
 } \
 </bib> ";


var sResult = objAltovaXML.XQuery.ExecuteAndGetResultAsString();
WScript.Echo(sResult);
```

C++

Im Folgenden sehen Sie eine C++-Codedatei, die zeigt, wie AltovaXML über die COM-Schnittstelle verwendet werden kann.

Codebeispiel

```
// TestAltovaXML.cpp : Defines the entry point for the console application.
//
#include "objbase.h"
#include <iostream>
#include "atlbase.h"

#import "AltovaXML_COM.exe" no_namespace raw_interfaces_only
// - or -
//#import "AltovaXML_COM.exe" raw_interfaces_only
//using namespace AltovaXMLLib;

int main(int argc, char* argv[])
```

```
{
	HRESULT hr = S_OK;
	hr = CoInitialize(NULL);
	if ( hr == S_OK )
	{
		IApplicationPtr ipApplication;

		hr = CoCreateInstance(
							__uuidof( Application
),
							NULL,
							CLSCTX_ALL,
							__uuidof(IApplication),

reinterpret_cast<void**>(&ipApplication)
						);
		if ( hr == S_OK )
		{
			IXQueryPtr ipXQuery;
			hr = ipApplication->get_XQuery( &ipXQuery );

			if ( hr == S_OK )
			{
				CComBSTR sXQExpr( "(1 to 10)[. mod 2 != 0]" );
				BSTR bstrResult;

				hr = ipXQuery->put_XQueryFromText( sXQExpr );
				hr = ipXQuery->ExecuteAndGetResultAsString(
&bstrResult );

				std::cout << (char*)_bstr_t(bstrResult) <<
std::endl;

				ipXQuery.Release();
			}

			ipApplication.Release();
		}

		CoUninitialize();
	}
	return 0;
}
```

 © 2012 Altova GmbH

2.3 Java-Schnittstelle

Sie können mit Hilfe von Java-Code auf die Applikations-API zugreifen. Damit Sie direkt vom Java-Code aus Zugriff auf den AltovaXML Automation Server haben, müssen sich die unten aufgelisteten Bibliotheken im `classpath` befinden. Sie sind im AltovaXML Applikationsordner im Ordner `JavaAPI` installiert.

- `AltovaAutomation.dll`: ein JNI Wrapper für Altova Automation Server
- `AltovaAutomation.jar`: Java-Klassen zum Aufrufen von Altova Automation Servern
- `AltovaXMLAPI.jar`: Java-Klassen, die als Wrapper für die XMLSpy Automation-Schnittstelle dienen
- `AltovaXMLAPI_JavaDoc.zip`: eine Javadoc Datei, die die Hilfedokumentation zur Java API enthält

Anmerkung: Um die Java API verwenden zu können, müssen sich die DLL und Jar-Dateien im Java Classpath befinden und `AltovaXML_COM.exe` muss als COM Server-Objekt registriert sein.

Java-Beispielprojekt
Im Lieferumfang Ihres Produkts ist ein Java-Beispielprojekt enthalten. Sie können das Java-Projekt nach Belieben testen und verwenden. Beachten Sie allerdings, dass der Ordner möglicherweise schreibgeschützt ist. In diesem Fall müssen Sie ihre Zugriffsberechtigung ändern. Nähere Informationen zum Java-Beispielprojekt finden Sie im Abschnitt Java-Beispielprojekt.

Frühere Schnittstelle
Ab Version 2012 wurde die frühere Java-Schnittstelle für AltovaXML durch eine neue ersetzt. Die Dokumentation zur alten Schnittstelle finden Sie hier: Alte Java API (nicht mehr verwendet).

Unterschiede zwischen der alten und der neuen Schnittstelle
Im Folgenden sind die Unterschiede zwischen der alten Schnittstelle (vor v2012) und der neuen Schnittstelle aufgelistet:

- Mit Hilfe der neuen `AltovaXMLFactory()` wird eine Instanz von `AltovaXMLFactory` anstelle von `AltovaXMLFactory.getInstance()` erstellt
- Die Methoden von `AltovaXMLFactory` zum Aufrufen des XML-Validators, des XQuery-, XSLT1- und XSLT2-Prozessors heißen `getXMLValidator()`, `getXQuery()` , `getXSLT1()`, `getXSLT2()`
- Um eine Prozessorinstanz der Factory freizugeben verwenden Sie `dispose()` anstelle von `releaseInstance()`
- Die Methode `enableJavaExtension` heißt in den verschiedenen Prozessorschnittstellen nun `setJavaExtensioEnabled`
- Die Methode `enableDotNetExtension` heißt in den verschiedenen Prozessorschnittstellen nun `setDotNetExtensioEnabled`

2.3.1 Java-Beispielprojekt

Das AltovaXML-Installationspaket enthält ein Java-Beispielprojekt, das Sie im Java-Ordner des Ordners API Examples finden: `AltovaXMLExamples/API/Java` im AltovaXML-Applikationsordner

Dieser Ordner enthält Java-Beispiele für die AltovaXML API. Sie können das Beispielprojekt mit Hilfe der Batch-Datei `BuildAndRun.bat` (im selben Ordner) direkt über die Befehlszeile testen oder Sie können es in Eclipse kompilieren und ausführen. Anleitungen dafür finden Sie weiter unten.

Dateiliste
Der Ordner für die Java-Beispiele enthält alle zum Ausführen des Beispielprojekts erforderlichen Dateien. Diese Dateien sind unten aufgelistet:

`AltovaAutomation.dll`	Java-COM Bridge: DLL-Teil
`AltovaAutomation.jar`	Java-COM Bridge: Java-Bibliotheksteil
`AltovaXMLAPI.jar`	Java-Klassen der AltovaXML API
`UseAltovaXML.java`	Java-Beispielquellcode
`BuildAndRun.bat`	Batch-Datei zum Kompilieren und Ausführen des Beispielcodes über die Befehlszeile. Es wird ein Ordner benötigt, in dem sich die Java Virtual Machine als Parameter befindet.
`.classpath`	Hilfedatei Eclipse-Projekt
`.project`	Eclipse-Projektdatei
`AltovaXMLAPI_JavaDoc.zip`	`Javadoc` Datei, die die Hilfedokumentation für die Java API enthält

Ausführen des Beispiels über die Befehlszeile
Um das Beispiel von der Befehlszeile aus auszuführen, öffnen Sie ein Eingabeaufforderungsfester, gehen Sie zum Ordner Java des Ordners API Examples (*Pfad siehe oben*) und geben Sie folgende Zeile ein:

```
buildAndRun.bat "<Path-to-the-Java-bin-folder>"
```

Der Java Binary-Ordner muss von einer JDK 1.5 oder höheren Version auf Ihrem Rechner sein.

Drücken Sie die **Eingabetaste.** Der Java-Quellcode in `UseAltovaXML.java` wird kompiliert und anschließend ausgeführt.

Laden des Beispiels in Eclipse
Öffnen Sie Eclipse und wählen Sie den Befehl **Import | Existing Projects into Workspace** um die Eclipse-Projektdatei (`.project`) im Ordner Java des Ordners API Examples (*Pfad siehe oben*) zu Eclipse hinzuzufügen. Daraufhin wird das Projekt`UseAltovaXML` in Ihrem Package Explorer oder Navigator angezeigt.

Wählen Sie das Projekt aus und klicken Sie anschließend auf **Run as | Java Application** um das Beispiel auszuführen.

Anmerkung: Sie können einen Klassennamen oder eine Methode der Java API auswählen

 © 2012 Altova GmbH

und F1 drücken, um Hilfe zu dieser Klasse oder Methode zu erhalten.

Java-Quellcode

Im Folgenden finden Sie den mit Kommentaren versehenen Java-Quellcode aus der Beispieldatei`UseAltovaXML.java`.

```
01 import com.altova.automation.AltovaXML.*;
02 import com.altova.automation.libs.AutomationException;
03
04
05 public class UseAltovaXML
06 {
07
08   /**
09    * @param args
10    */
11   public static void main(String[] args)
12   {
13     // Locate samples installed with the product.
14     // REMARK: You will need to modify this if you use a different major version, use
the 64-bit
15     //         version of the program, or installed AltovaXML in a different
location.
16     String strExamplesFolder = System.getenv("ProgramFiles") +
"/Altova/AltovaXML2012/AltovaXMLExamples/";
17
18     String inFilename = strExamplesFolder + "simple.xml";
19     String xqFilename = strExamplesFolder + "CopyInput.xq";
20     System.out.println("AltovaXML Java JNI XQuery");
21
22     AltovaXMLFactory xmlFactory = null;
23     try
24     {
25       // Get application instance
26       xmlFactory = new AltovaXMLFactory();
27
28       // Get XML validator and XQ method pointers from the application instance
29       XMLValidator validator = xmlFactory.getXMLValidator();
30       XQuery xQuery = xmlFactory.getXQuery();
31
32       // We only want to work with input files that are well-formed.
33       validator.setInputXMLFileName(inFilename);
34       if (validator.isWellFormed())
35       {
36         // If the file is well-formed, copy it using XQuery
37         xQuery.setInputXMLFileName(inFilename);
38         xQuery.setXQueryFileName(xqFilename);
39
40         // Test return value
41         String resultString = xQuery.executeAndGetResultAsString();
42         if (resultString == null )
43               System.out.println("XQuery error: " + xQuery.getLastErrorMessage());
44         else
45               System.out.println("Transform contents: " + resultString );
46       }
47       else
48         System.out.println("Not wellformed error: " + validator.getLastErrorMessage()
);
49
50     }
51     catch (AutomationException e)
52     {
53       // An error occurred when talking to the AltovaXML COM interface.
54       System.out.println("Error accessing AltovaXML: " + e.getMessage());
55     }
56     finally
57     {
```

```
58        // Now we can release the factory to immediately shut-down AltovaXML_COM.exe
59        if ( xmlFactory != null )
60          xmlFactory.dispose();
61      }
62    }
63  }
```


© 2012 Altova GmbH

2.3.2 Klassen

Im Folgenden sehen Sie eine Übersicht über die Klassen von `com.altova.engines`. Ausführliche Beschreibungen dazu finden Sie in den entsprechenden Abschnitten.

- `AltovaXMLFactory`
  Erstellt über einen nativen Aufruf eine neue AltovaXML COM Serverobjekt-Instanz und bietet Zugriff auf die AltovaXML Prozessoren.
- `XMLValidator`
  Klasse, die den XMLValidator enthält.
- `XQuery`
  Klasse, die den XQuery 1.0-Prozessor enthält.
- `XSLT1`
  Klasse, die den XSLT 1.0-Prozessor enthält.
- `XSLT2`
  Klasse, die den XSLT 2.0-Prozessor enthält.

AltovaXMLFactory

```
Öffentliche Klasse AltovaXMLFactory
erweitert java.lang.Object
```

Beschreibung
Mit Hilfe von `AltovaXMLFactory()` können Sie eine neue AltovaXML COM Serverobjektinstanz erstellen, über die Sie Zugriff auf die AltovaXML-Prozessoren erhalten. Die Beziehung zwischen `AltovaXMLFactory` und dem AltovaXML COM-Objekt ist eine 1:1-Beziehung, d.h., dass nachfolgende Aufrufe der Funktion `getENGINENAME()` Schnittstellen für dieselbe Prozessorinstanz zurückgeben.

Methoden
Es sind die folgenden Methoden definiert. Mit Hilfe der Methoden für Kataloge können Kataloge für den Dokumenten-Lookup hinzugefügt werden. Nachdem Kataloge hinzugefügt wurden, werden Sie so lange für Lookup-Funktionen verwendet, bis der COM Server beendet wird. Hinzugefügte Kataloge können nicht entfernt werden.

`addXMLCatalogDefault()`
`public void addXMLCatalogDefault()`
Fügt den Altova-Standardkatalog `RootCatalog.xml` zu den Katalogen hinzu.

`addXMLCatalogFromFile( string catalogfilename )`
`public void addXMLCatalogFromFile(java.lang.String bstrXMLCatalogFileName)`
Fügt, den durch `catalogfilename` identifizierten Katalog zu den Katalogen hinzu.

`addXMLCatalogFromText( string catalogtext )`
`public void addXMLCatalogFromText(java.lang.String bstrXMLCatalogText)`
Fügt den Katalog mit dem Inhalt `catalogtext` zu den Katalogen hinzu.

`dispose`
`public void dispose()`
Gibt die Verbindung des Objekts mit dem COM-Server frei.

getXMLValidator
`public XMLValidator getXMLValidator()`
Ruft den XMLValidator auf. Die Verbindung des Objekts mit dem Prozessor muss nach der Verwendung mit Hilfe der Methode `dispose()` wieder freigegeben werden.
Rückgabewert:
eine neue `XMLValidator` Instanz dieser AltovaXMLFactory-Klasse.

getXQuery
`public XQuery getXQuery()`
Ruft den XQuery-Prozessor auf. Die Verbindung des Objekts mit dem Prozessor muss nach der Verwendung mit Hilfe der Methode `dispose()` wieder freigegeben werden.
Rückgabewert:
eine neue `XQuery 1.0 engine` Instanz dieser AltovaXMLFactory-Klasse.

getXSLT1
`public XSLT1 getXSLT1()`
Ruft den XSLT 1.0-Prozessor auf. Die Verbindung des Objekts mit dem Prozessor muss nach der Verwendung mit Hilfe der Methode `dispose()` wieder freigegeben werden.
Rückgabewert:
eine neue `XSLT 1.0 engine` Instanz dieser AltovaXMLFactory-Klasse.

getXSLT2
`public XSLT2 getXSLT2()`
Ruft den XSLT 2.0-Prozessor auf. Die Verbindung des Objekts mit dem Prozessor muss nach der Verwendung mit Hilfe der Methode `dispose()` wieder freigegeben werden.
Rückgabewert:
eine neue `XSLT 2.0 engine` Instanz dieser AltovaXMLFactory-Klasse.

XMLValidator

```
Öffentliche Klasse AXMLValidator
erweitert java.lang.Object
```

Beschreibung

Die Klasse, die den XMLValidator enthält. Keine direkte Konstruktion/kein direkter Zugriff möglich. Ermöglicht durch Aufruf der Funktion `getXMLValidator()` in einer Instanz von `AltovaXMLFactory` Zugriff auf den XML-Validator.

Methoden

Es sind die folgenden Methoden definiert.

getLastErrorMessage
`public java.lang.String getLastErrorMessage()`
Ruft die letzte Fehlermeldung des Prozessors ab.

 © 2012 Altova GmbH

Rückgabewert:
ein String, der die letzte Fehlermeldung enthält.

isValid
`public boolean` **`isValid`**`()`
Validiert die XML-Input-Daten anhand des darin angegebenen Schemas/der DTD.
Rückgabewert:
`true` bei Erfolg, `false` bei Fehlschlag. Falls die Validierung fehlschlägt, steht die Methode `getLastErrorMessage()` zur Verfügung.

isValidWithExternalSchemaOrDTD
`public boolean` **`isValidWithExternalSchemaOrDTD`**`()`
Validiert die XML-Input-Daten anhand der externen DTD/des externen Schemas, die/das mit den Funktionen `setDTDFileName()`, `setDTDFromText()`, `setSchemaFileName()`, and `setSchemaFromText()` definiert werden kann. *Eine Beschreibung dieser Methoden finden Sie weiter unten.*
Rückgabewert:
`true` bei Erfolg, `false` bei Fehlschlag. Falls die Validierung fehlschlägt, steht die Methode `getLastErrorMessage()` zur Verfügung.

isWellFormed
`public boolean` **`isWellFormed`**`()`
Überprüft die XML-Input-Daten auf Wohlgeformtheit.
Rückgabewert:
`true` bei Erfolg, `false` bei Fehlschlag. Falls die Prüfung fehlschlägt, steht die Methode `getLastErrorMessage()` zur Verfügung.

dispose
`public void` **`dispose`**`()`
Gibt die Verbindung des Objekts mit dem COM-Server frei.

setDTDFileName
`public void` **`setDTDFileName`**`(java.lang.String str)`
Definiert den Dateinamen der externen DTD.
Parameter:
`str`: eine absolute URL, die den Basispfad der DTD angibt.

setDTDFromText
`public void` **`setDTDFromText`**`(java.lang.String str)`
Definiert den Textwert der externen DTD.
Parameter:
`str`: ein String, der die DTD als Text enthält.

setInputXMLFileName
`public void` **`setInputXMLFileName`**`(java.lang.String str)`
Definiert den Dateinamen der XML-Input-Daten. Beachten Sie, dass Sie absolute URLs verwenden müssen.
Parameter:

str: eine absolute URL, die den Basispfad der XML-Daten angibt.

setInputXMLFromText
public void **setInputXMLFromText**(java.lang.String str)
Definiert den Textwert für die XML-Input-Daten. Beispiel: setInputXMLFromText("<doc><a>text </doc>")
Parameter:
str: ein String, der XML-Daten enthält.

setSchemaFileName
public void **setSchemaFileName**(java.lang.String str)
Definiert den Dateinamen eines externen Schemas.
Parameter:
str: eine absolute URL, die den Basispfad des Schemas angibt.

setSchemaFromText
public void **setSchemaFromText**(java.lang.String str)
Definiert den Textwert für ein externes Schema.
Parameter:
str: ein String, der ein Schema als Text enthält.

setTreatXBRLInconsistenciesAsErrors
public void **setTreatXBRLInconsistenciesAsErrors**(boolean param)
Definiert einen Boolschen Wert, mit dem festgelegt wird, ob XBRL-Inkonsistenzen als Fehler behandelt werden sollen oder nicht.
Parameter:
param: ein Boolscher Wert.

XQuery

Öffentliche Klasse XQuery
erweitert java.lang.Object

Beschreibung
Klasse, die den XQuery 1.0-Prozessor enthält. Keine direkte Konstruktion/kein direkter Zugriff möglich. Erlaubt durch Aufruf der Funktion getXQuery() in einer Instanz von AltovaXMLFactory Zugriff auf den XQuery-Prozessor.

Methoden
Es sind die folgenden Methoden definiert.

addExternalVariable
public void **addExternalVariable**(java.lang.String strName,
java.lang.String strVal)
Fügt den Namen und Wert einer externen Variablen hinzu.
Parameter:
strName: ein String, der als Variablennamen einen gültigen QName enthält.
strVal: ein String, der den Wert der Variablen enthält; dieser Wert wird als String verwendet.

 © 2012 Altova GmbH

addExternalVariableAsXPath
```
public void addExternalVariableAsXPath(java.lang.String strName,
                                       java.lang.String strVal)
```
Fügt den Namen und Wert einer externen Variablen hinzu, wobei der Wert als XPath 2.0-Ausdruck ausgewertet wird.
Parameter:
`strName`: ein String, der als Variablennamen einen gültigen QName enthält.
`strVal`: ein String, der den Wert der Variablen enthält; dieser Wert wird als XPath 2.0-Ausdruck ausgewertet.

clearExternalVariableList
```
public void clearExternalVariableList()
```
Löscht die Liste der externen Variablen.

dispose
```
public void dispose()
```
Gibt die Verbindung des Objekts mit dem COM-Server frei.

execute
```
public boolean execute(java.lang.String sOutFile)
```
Führt die Datei aus und speichert das Ergebnis in einer Datei. Falls ein Fehler auftritt, können Sie die Methode `getLastErrorMessage()` verwenden.
Parameter:
`sOutFile`: eine absolute URL mit dem Pfad der Ausgabedatei.
Rückgabewert:
`true`
Anmerkung: Wenn kein Ausgabeparameter angegeben ist, ist der Rückgabetyp `void`:
```
public void execute(java.lang.String sOutFile)
```

executeAndGetResultAsString
```
public java.lang.String executeAndGetResultAsString()
```
Führt die Datei aus und gibt das Ergebnis als UTF-16 Textstring zurück.Falls ein Fehler auftritt, können Sie die Methode `getLastErrorMessage()` verwenden.
Rückgabewert:
ein String, der das serialisierte Ergebnis enthält. Bei einem Fehler wird ein leerer String zurückgegeben.

getLastErrorMessage
```
public java.lang.String getLastErrorMessage()
```
Ruft die letzte Fehlermeldung vom Prozessor ab.
Rückgabewert:
ein String, der die letzte Fehlermeldung enthält.

getOutputEncoding
```
public java.lang.String getOutputEncoding()
```
Ruft die für das Ergebnisdokument definierte Kodierung ab.
Rückgabewert:
ein String, der den Namen der Kodierung enthält.

getOutputIndent
`public boolean` **`getOutputIndent`**`()`
Ruft die für das Ergebnisdokument definierte Einrückung der Ausgabe ab.
Rückgabewert:
der aktuelle Wert des Einrückungsserialisierungsparameters.

getOutputMethod
`public java.lang.String` **`getOutputMethod`**`()`
Ruft die Serialisierungsmethdoe für das Ergebnisdokument ab.
Rückgabewert:
die aktuelle Serialisierungsmethode.

getOutputOmitXMLDeclaration
`public boolean` **`getOutputOmitXMLDeclaration`**`()`
Ruft den Wert der für das Ergebnisdokument definierten Option `omitXMLDeclaration` ab.
Rückgabewert:
ein Boolscher Wert des Parameters `omit-xml-declaration`.

setDotNetExtensionsEnabled
`public void` **`enableDotNetExtensions`**`(boolean bEnable)`
Aktiviert/Deaktiviert .NET-Erweiterungsfunktionen.

setInputXMLFileName
`public void` **`setInputXMLFileName`**`(java.lang.String str)`
Definiert den Dateinamen für die XML-Input-Daten. Beachten Sie, dass absolute URLs verwendet werden müssen
Parameter:
`str`: eine absolute URL, die den Basispfad der XML-Daten angibt.

setInputXMLFromText
`public void` **`setInputXMLFromText`**`(java.lang.String str)`
Definiert den Textwert für die XML-Input-Daten. Beispiel: `setInputXMLFromText( "<doc> <a>text</a> </doc>" )`.
Parameter:
`str`: ein String, der XML-Daten enthält.

setJavaExtensionsEnabled
`public void` **`enableJavaExtensions`**`(boolean bEnable)`
Aktiviert/Deaktiviert Java-Erweiterungsfunktionen.

setOutputEncoding
`public void` **`setOutputEncoding`**`(java.lang.String str)`
Definiert die Kodierung für das Ergebnisdokument.
Parameter:
`str`: ein String, der einen Kodierungsnamen enthält (z.B: `UTF-8`, `UTF-16`, `ASCII`, `8859-1`,

 © 2012 Altova GmbH

1252)

setOutputIndent
public void **setOutputIndent**(boolean bVal)
Aktiviert/Deaktiviert die Einrückungsoption für das Ergebnisdokument.
Parameter:
bVal: ein Boolscher Wert zum Aktivieren/Deaktivieren der Einrückung.

setOutputMethod
public void **setOutputMethod**(java.lang.String str)
Definiert die Serialisierungsmethode für ein Ergebnisdokument.
Parameter:
str: ein String, der die Serialisierungsmethode enthält. Gültige Werte: xml, xhtml, html, text.

setOutputOmitXMLDeclaration
public void **setOutputOmitXMLDeclaration**(boolean bVal)
Aktiviert/Deaktiviert die Serialisierungsoption omitXMLDeclaration für das Ergebnisdokument.
Parameter:
bVal: ein neuer Boolscher Wert für den Parameter omit-xml-declaration.

setXQueryFileName
public void **setXQueryFileName**(java.lang.String str)
Definiert den Dateinamen des XQuery-Dokuments.
Parameter:
str: eine absolute URL mit dem Basispfad der XQuery-Datei.

setXQueryFromText
public void **setXQueryFromText**(java.lang.String str)
Defniert den Textwert für die XQuery-Anweisung.
Parameter:
str: ein String, der die XQuery-Anweisung enthält.

XSLT1

Öffentliche Klasse XSLT1
erweitert java.lang.Object

Beschreibung

Klasse, die den XSLT 1.0-Prozessor enthält. Keine direkte Konstruktion/kein direkter Zugriff möglich. Erlaubt durch Aufruf der Funktion getXSLT1() in einer Instanz von IAltovaXMLFactory Zugriff auf den XSLT 1.0-Prozessor.

Methoden

Es sind die folgenden Methoden definiert.

addExternalParameter
```
public void addExternalParameter(java.lang.String strName,
                                 java.lang.String strVal)
```
Fügt den Namen und Wert eines externen Parameters hinzu.
Parameter:
`strName`: ein String, der einen gültigen QName als Parameternamen enthält.
`strVal`: ein String, der den Wert des Parameters enthält; dieser Wert wird als XPath-Ausdruck ausgewertet.

clearExternalParameterList
```
public void clearExternalParameterList()
```
Löscht die Liste der externen Parameter.

dispose
```
public void dispose()
```
Gibt die Verbindung des Objekts mit dem COM-Server frei.

execute
```
public boolean execute(java.lang.String sOutFile)
```
Führt die Datei aus und speichert das Ergebnis in einer Datei. Falls ein Fehler auftritt, können Sie die Methode `getLastErrorMessage()` verwenden.
Parameter:
`sOutFile`: eine absolute URL mit dem Pfad der Ausgabedatei.
Rückgabewert:
`true` bei Erfolg, `false` bei einem Fehler.
Anmerkung: Wenn kein Ausgabeparameter angegeben ist, ist der Rückgabetyp `void`:
```
public void execute(java.lang.String sOutFile)
```

executeAndGetResultAsString
```
public java.lang.String executeAndGetResultAsString()
```
Führt die Datei aus und gibt das Ergebnis als UTF-16 Textstring zurück.Falls ein Fehler auftritt, können Sie die Methode `getLastErrorMessage()` verwenden.
Rückgabewert:
ein String, der das serialisierte Ergebnis enthält. Bei einem Fehler wird ein leerer String zurückgegeben.

getLastErrorMessage
```
public java.lang.String getLastErrorMessage()
```
Ruft die letzte Fehlermeldung vom Prozessor ab.
Rückgabewert:
ein String, der die letzte Fehlermeldung enthält.

setDotNetExtensionsEnabled
```
public void enableDotNetExtensions(boolean bEnable)
```
Aktiviert/Deaktiviert .NET-Erweiterungsfunktionen.

© 2012 Altova GmbH

setInputXMLFileName
`public void` **`setInputXMLFileName`**`(java.lang.String str)`
Definiert den Dateinamen für die XML-Input-Daten. Beachten Sie, dass absolute URLs verwendet werden müssen.
Parameter:
`str`: eine absolute URL mit dem Basispfad der XML-Daten.

setInputXMLFromText
`public void` **`setInputXMLFromText`**`(java.lang.String str)`
Definiert den Textwert für die XML-Input-Daten. Beispiel: `setInputXMLFromText( "<doc> <a>text</a> </doc>" ).`
Parameter:
`str`: ein String, der XML-Daten enthält.

setJavaExtensionsEnabled
`public void` **`enableJavaExtensions`**`(boolean bEnable)`
Aktiviert/Deaktiviert Java-Erweiterungsfunktionen.

setXSLFileName
`public void` **`setXSLTFileName`**`(java.lang.String str)`
Definiert den Dateinamen für die XSLT-Daten.
Parameter:
`str`: eine absolute URL mit dem Basispfad der XSLT-Daten

setXSLFromText
`public void` **`setXSLTFromText`**`(java.lang.String str)`
Definiert den Textwert für die XSLT-Daten.
Parameter:
`str`: ein String, der die serialisierten XSLT-Daten enthält.

setXSLStackSize
`public void` **`setXSLTStackSize`**`(long nVal)`
Die Stapelgröße ist die maximale Tiefe ausgeführter Anweisungen. Bei Überschreitung der Stapelgröße bei einer Transformation wird eine Fehlermeldung ausgegeben.
Parameter:
`nVal`: numerischer Wert für eine neue Stapelgröße. Der Wert muss größer als 100 sein. Der Anfangswert ist 1000.

XSLT2

```
Öffentliche Klasse XSLT2
erweitert java.lang.Object
```

Beschreibung
Klasse, die den XSLT 2.0-Prozessor enthält. Keine direkte Konstruktion/kein direkter Zugriff möglich. Ermöglicht durch Aufruf der Funktion `getXSLT2()` in einer Instanz von `AltovaXMLFactory` den Zugriff auf den XSLT 2.0-Prozessor. Beachten Sie: Der XSLT 2.0-Prozessor kann zur Verarbeitung eines XSLT 1.0 Stylesheet im

Rückwärtskompatibilitätsmodus verwendet werden. Die Ausgabe kann sich allerdings von der eines XSLT 1.0-Prozessors, der dasselbe XSLT 1.0 Stylesheet verarbeitet, unterscheiden.

Methoden
Es sind die folgenden Methoden definiert.

addExternalParameter
```
public void addExternalParameter(java.lang.String strName,
                                 java.lang.String strVal)
```
Fügt den Namen und Wert eines externen Parameters hinzu.
Parameter:
`strName`: ein String, der einen gültigen QName als Parameternamen enthält.
`strVal`: ein String, der den Wert des Parameters enthält; dieser Wert wird als XPath-Ausdruck ausgewertet.

clearExternalParameterList
```
public void clearExternalParameterList()
```
Löscht die Liste der externen Parameter.

dispose
```
public void dispose()
```
Gibt die Verbindung des Objekts mit dem COM-Server frei.

execute
```
public boolean execute(java.lang.String sOutFile)
```
Führt die Datei aus und speichert das Ergebnis in einer Datei. Falls ein Fehler auftritt, können Sie die Methode `getLastErrorMessage()` verwenden.
Parameter:
`sOutFile`: eine absolute URL mit dem Pfad der Ausgabedatei.
Rückgabewert:
`true` bei Erfolg, `false` bei einem Fehler.
Anmerkung: Wenn kein Ausgabeparameter angegeben ist, ist der Rückgabetyp `void`:
```
public void execute(java.lang.String sOutFile)
```

executeAndGetResultAsString
```
public java.lang.String executeAndGetResultAsString()
```
Führt die Datei aus und gibt das Ergebnis als UTF-16 Textstring zurück.Falls ein Fehler auftritt, können Sie die Methode `getLastErrorMessage()` verwenden.
Rückgabewert:
ein String, der das serialisierte Ergebnis enthält. Bei einem Fehler wird ein leerer String zurückgegeben.

getLastErrorMessage
```
public java.lang.String getLastErrorMessage()
```
Ruft die letzte Fehlermeldung vom Prozessor ab.
Rückgabewert:
ein String, der die letzte Fehlermeldung enthält.

 © 2012 Altova GmbH

setDotNetExtensionsEnabled
`public void` **`enableDotNetExtensions`**`(boolean bEnable)`
Aktiviert/Deaktiviert .NET-Erweiterungsfunktionen.

setInitialTemplateMode
`public void` **`setInitialTemplateMode`**`(java.lang.String str)`
Definiert den Anfangsvorlagenmodus für die Transformation.

setInitialTemplateName
`public void` **`setInitialTemplateName`**`(java.lang.String str)`
Definiert den Anfangsvorlagennamen für die Transformation.

setInputXMLFileName
`public void` **`setInputXMLFileName`**`(java.lang.String str)`
Definiert den Dateinamen für die XML-Input-Daten. Beachten Sie, dass absolute URLs verwendet werden müssen.
Parameter:
`str`: eine absolute URL mit dem Basispfad der XML-Daten.

setInputXMLFromText
`public void` **`setInputXMLFromText`**`(java.lang.String str)`
Definiert den Textwert für die XML-Input-Daten. Beispiel: `setInputXMLFromText( "<doc> <a>text</a> </doc>" ).`
Parameter:
`str`: ein String, der XML-Daten enthält.

setJavaExtensionsEnabled
`public void` **`enableJavaExtensions`**`(boolean bEnable)`
Aktiviert/Deaktiviert Java-Erweiterungsfunktionen.

setXSLFileName
`public void` **`setXSLTFileName`**`(java.lang.String str)`
Definiert den Dateinamen für die XSLT-Daten.
Parameter:
`str`: eine absolute URL mit dem Basispfad der XSLT-Daten

setXSLFromText
`public void` **`setXSLTFromText`**`(java.lang.String str)`
Definiert den Textwert für die XSLT-Daten.
Parameter:
`str`: ein String, der die serialisierten XSLT-Daten enthält.

setXSLStackSize
`public void` **`setXSLTStackSize`**`(long nVal)`
Die Stapelgröße ist die maximale Tiefe ausgeführter Anweisungen. Bei Überschreitung der Stapelgröße bei einer Transformation wird eine Fehlermeldung ausgegeben.
Parameter:
`nVal`: numerischer Wert für eine neue Stapelgröße. Der Wert muss größer als 100 sein. Der

Anfangswert ist 1000.

© 2012 Altova GmbH

2.3.3 Alte Java API (nicht mehr verwendet)

Die in diesem Abschnitt (Alte Java API) beschriebenen Objekte werden ab v2012 nicht mehr verwendet.

Informationen zum Aufrufen der Application API von Java-Code aus finden Sie im Abschnitt: Java-Schnittstelle.

Über die AltovaXML Java-Schnittstelle (`AltovaXML.jar`) wird mittels Funktionen in der `AltovaXMLLib.dll` eine Verbindung zur AltovaXML COM-Schnittstelle hergestellt. Diese DLL wird bei der Installation von AltovaXML mit dem AltovaXML Installer im Verzeichnis `WINDIR\system32\` installiert. `AltovaXML.jar` enthält das Paket `com.altova.engines`, welches die Altova-Prozessoren enthält.

Installation
Um die Java-Schnittstelle verwenden zu können, muss die Datei `AltovaXML.jar` zur `CLASSPATH`. COM Registrierung hinzugefügt werden. Dies geschieht automatisch bei der Installation. Wenn Sie den Pfad zur Datei `AltovaXML_COM.exe` nach der Installation ändern, müssen Sie AltovaXML durch Ausführen des Befehls `AltovaXML_COM.exe /regserver` als COM-Serverobjekt registrieren. Nähere Informationen dazu finden Sie unter Registrieren von AltovaXML als COM Serverobjekt.

Dokumentation
Dieser Abschnitt enthält eine ausführliche Beschreibung der AltovaXML Java-Schnittstelle und ist auch im HTML-Format im ZIP-Archiv `AltovaXMLJavaDocs.zip` , welches Sie im `AltovaXML2013` Anwendungsordner finden, verfügbar.

Beispiele
Nähere Beispiele finden Sie in den Beispieldateien im Applikationsordner im Ordner `AltovaXMLExamples`.

Das `com.altova.engines` Paket
Zur Verwendung der Java-Schnittstelle benötigen Sie das Paket `com.altova.engines`. Dies ist die Java-Schnittstelle für das AltovaXML COM Serverobjekt; sie bietet Zugriff auf den XML Validator und den XSLT 1.0-, den XSLT 2.0- und den XQuery 1.0-Prozessor.

Das Paket `com.altova.engines` stellt über die nativen Funktionen in der unter `WINDIR\system32\directory` installierten Datei `AltovaXMLLib.dll` die Verbindung mit der AltovaXML COM Schnittstelle her.

Um eine Verbindung mit einer neuen Instanz des AltovaXML COM Serverobjekts herzustellen, verwenden Sie die statische Methode `getInstance()` der Klasse `AltovaXMLFactory`. Aus der Schnittstelle, die zurückgegeben wird, können Sie mittels der Funktion `getENGINENAMEInstance()` den gewünschten Prozessor auswählen.

Im Folgenden finden Sie ein Codebeispiel, bei dem die Java-Schnittstelle verwendet wird:

```
import com.altova.engines.*;

/**
```

```
 * Test application for AltovaXML COM components java interface
 */
public class AltovaXMLTest {
    /**
     * public constructor for AltovaXMLTest
     */
    public AltovaXMLTest(){
    }

    /**
     * application main
     */
    public static void main(String[] args) {
        System.out.println("AltovaXML Java Interface Test Application");

        //request a COM server object - fails if AltovaXML is not registered
        IAltovaXMLFactory objXmlApp = AltovaXMLFactory.getInstance();

        if ( objXmlApp != null ) {
            //get interface for the XQuery engine
            IXQuery xquery = objXmlApp.getXQueryInstance();
            //set XQuery statement
            xquery.setXQueryStatement("<doc><a>{1 to 3}</a>This data is
well-formed.</doc>");
            //execute the statement previously set.
            //There was no input XML specified so the initial context is
empty.
            String sres = xquery.executeAndGetResultAsString();
            //release XQuery engine's connection to the COM server object
            xquery.releaseInstance();
            System.out.println(sres);


            IXMLValidator validator = objXmlApp.getXMLValidatorInstance();
            validator.setInputXMLFromText(sres);
            boolean b = validator.isWellFormed();
            if ( b )
                System.out.println("XML data is well-formed.");
            else
                System.out.println("Data is not well-formed.");
            validator.releaseInstance();

            //release Application object connection to the COM server object.
            //After this the COM server object will shut down automatically.
            objXmlApp.releaseInstance();
        } else{
            System.out.println("Creating instance of IAltovaXMLFactory
failed.");
            System.out.println("Please make sure AltovaXML.exe is correctly
registered!");
        }
    }
}
```

Beispiel

Mit dem unten aufgelisteten Code wird überprüft, ob die vorliegende XML-Datei wohlgeformt ist und anschließend wird ein XQuery-Dokument ausgeführt.

Um eine Verbindung zu einer neuen Instanz des AltovaXML COM Serverobjekts herzustellen, verwenden Sie die statische Methode `getInstance()` der Klasse `AltovaXMLFactory`. Aus der zurückgegebenen Schnittstelle können Sie mit Hilfe der Funktion `getENGINENAMEInstance()` den gewünschten Prozessor auswähle (z.B.: `getXMLValidatorInstance()`).

 © 2012 Altova GmbH

```
            // Locate samples installed with the product.
            // REMARK: You will need to modify this if you use a different
major version.
            String strExamplesFolder = System.getenv("ProgramFiles") +
"/Altova/AltovaXML2011/AltovaXMLExamples/";

            String inFilename = strExamplesFolder + "simple.xml";
            String xqFilename = strExamplesFolder + "CopyInput.xq";
            System.out.println("AltovaXML Java JNI XQuery");

            try
            {
                  // get application instance
                  IAltovaXMLFactory objXmlApp = AltovaXMLFactory.getInstance
();

                  // get XML Validator and XQ method pointers from the
application instance
                  IXMLValidator validator =
objXmlApp.getXMLValidatorInstance();
                  IXQuery xQuery = objXmlApp.getXQueryInstance();

                  // remove comments on line below to see error being caught*/
                  validator.setInputXMLFileName(inFilename);
                  if (validator.isWellFormed())
                  {
                        // if the file is well-formed copy it using XQuery
                        xQuery.setInputXMLFileName(inFilename);
                        xQuery.setXQueryFileName(xqFilename);

                        // test return value
                        String resultString =
xQuery.executeAndGetResultAsString();
                        if (resultString == null )
                              System.out.println("XQuery error: " +
xQuery.getLastErrorMessage());
                        else
                              System.out.println("Transform contents: " +
resultString );
                  }
                  else
                        System.out.println("Not wellformed error: " +
validator.getLastErrorMessage() );

                  // release instance pointer
                  objXmlApp.releaseInstance();
            }
            catch (Exception e)
            {
                  System.out.println("Error: " + e);
            }
      }
```

Beispiele
Weitere Beispiele finden Sie in den Beispieldateien im Ordner `AltovaXMLExamples` des AltovaXML-Applikationsordners.

Schnittstellen

Im Folgenden sehen Sie eine Übersicht über die Schnittstellen von `com.altova.engines`. Ausführliche Beschreibungen finden Sie in den jeweiligen Abschnitten.

- IAltovaXMLEngine
  Ausgangsschnittstelle für XML Validator sowie den XSLT 1.0-, XSLT 2.0- und XQuery 1.0-Prozessor.
- IAltovaXMLFactory
  Schnittstelle für AltovaXML COM Object Wrapper.
- IExecutable
  Executable-Schnittstelle für Prozessoren.
- IReleasable
  Schnittstelle für Release-Funktionalität.
- IXMLValidator
  Schnittstelle für XML Validator.
- IXQuery
  Schnittstelle für den XQuery 1.0-Prozessor.
- IXSLT
  Schnittstelle für die XSLT-Prozessoren.

IAltovaXMLEngine

Ausgangsschnittstelle für den XMLValidator und den XSLT 1.0-, XSLT 2.0- und XQuery-Prozessor. Öffentliche Schnittstelle, die IReleasable erweitert.

Superschnittstelle: IReleasable
Subschnittstelle: XMLValidator, IXQuery, IXSLT
Implementierende Klassen: XMLValidator, XQuery, XSLT1, XSLT2

Methoden
Es sind die folgenden Methoden definiert.

setInputXMLFileName
`public void setInputXMLFileName(java.lang.String filename)`
Definiert den Dateinamen für die XML-Input-Daten. Bitte beachten Sie, dass Sie absolute URLs verwenden müssen.
Parameter:
`filename`: eine absolute URL, die den Basispfad zu den XML-Daten angibt.

setInputXMLFromText
`public void setInputXMLFromText(java.lang.String text)`
Definiert den Textwert für die XML-Inputdaten. Z.B.: `setInputXMLFromText( "<doc> <a>text</a> </doc>" )`
Parameter:
`text`: ein String, der XML-Daten enthält.

getLastErrorMessage
`public java.lang.String getLastErrorMessage()`
Ruft die letzte Fehlermeldung vom Prozessor ab.
Rückgabewert:
ein String, der die letzte Fehlermeldung enthält.

 © 2012 Altova GmbH

IAltovaXMLFactory

Schnittstelle für AltovaXML COM Object Wrapper. Bietet Zugriff auf die Schnittstellen des XMLValidators, des XSLT 1.0-, XSLT 2.0- und XQuery 1.0-Prozessors. Öffentliche Schnittstelle, die IReleasable erweitert.

Superschnittstelle: IReleasable
Implementierende Klassen: AltovaXMLFactory

Methoden
Es sind die folgenden Methoden definiert.

getXQueryInstance
public IXQuery **getXQueryInstance**()
Erstellt eine neue Instanz der XQuery-Klasse für die aktuelle XQuery-Prozessor-Instanz. Die Verbindung des Objekts mit dem Prozessor muss nach der Verwendung wieder freigegeben werden. Verwenden Sie dazu die in der IReleasable Schnittstelle deklarierte Funktion releaseInstance().
Rückgabewert:
die IXQuery Schnittstelle der neu erstellten Klasse.

getXSLT1Instance
public IXSLT **getXSLT1Instance**()
Erstellt eine neue Instanz der XSLT1-Klasse für die aktuelle XSLT 1.0-Prozessor-Instanz. Die Verbindung des Objekts mit dem Prozessor muss nach der Verwendung wieder freigegeben werden. Verwenden Sie dazu die in der IReleasable Schnittstelle deklarierte Funktion releaseInstance().
Rückgabewert:
die IXSLT Schnittstelle der neu erstellten Klasse.

getXSLT2Instance
public IXSLT **getXSLT2Instance**()
Erstellt eine neue Instanz der XSLT2-Klasse für die aktuelle XSLT 2.0-Prozessor-Instanz. Die Verbindung des Objekts mit dem Prozessor muss nach der Verwendung wieder freigegeben werden. Verwenden Sie dazu die in der IReleasable Schnittstelle deklarierte Funktion releaseInstance().
Rückgabewert:
die IXSLT Schnittstelle der neu erstellten Klasse.

getXMLValidatorInstance
public IXMLValidator **getXMLValidatorInstance**()
Erstellt eine neue Instanz der XMLValidator-Klasse für die aktuelle XML Validator-Instanz. Die Verbindung des Objekts mit dem Prozessor muss nach der Verwendung wieder freigegeben werden. Verwenden Sie dazu die in der IReleasable Schnittstelle deklarierte Funktion releaseInstance().
Rückgabewert:
die IXMLValidator Schnittstelle der neu erstellten Klasse.

Mit den folgenden Methoden können Kataloge für Dokument-Lookup-Zwecke hinzugefügt werden. Nachdem die Kataloge hinzugefügt wurden, werden sie so lange zu Lookup Zwecken

herangezogen, bis der COM Server beendet wird. Hinzugefügte Kataloge können nicht mehr entfernt werden.

app.AddXMLCatalogDefault()
Fügt den Altova-Standardkatalog `RootCatalog.xml` zu den Katalogen hinzu.

app.AddXMLCatalogFromFile(string catalogfilename)
Fügt den durch `catalogfilename` identifizierten Katalog zu den Katalogen hinzu.

app.AddXMLCatalogFromText(string catalogtext)
Fügt den Katalog mit dem Inhalt `catalogtext` zu den Katalogen hinzu.

IExecutable

Executable-Schnittstelle für Prozessoren. Öffentliche Schnittstelle.

Subschnittstelle: IXQuery, IXSLT
Implementierende Klassen: XQuery, XSLT1, XSLT2

Methoden
Es sind die folgenden Methoden definiert.

execute
`public boolean` **execute**`(java.lang.String outfilename)`
Führt das Ergebnis aus und speichert es in einer Datei. Bei einem Fehler können Sie die in IAltovaXMLEngine deklarierte Funktion getLastErrorMessage() verwenden, um zusätzliche Informationen abzurufen.
Parameter:
`outfilename`: eine absolute URL, die den Pfad der Ausgabedatei angibt.
Rückgabewert:
`true` bei erfolgreicher Ausführung, `false` bei Fehler.

executeAndGetResultAsString
`public java.lang.String` **executeAndGetResultAsString**`()`
Führt das Ergebnis aus und speichert es als String.Bei einem Fehler können Sie die in IAltovaXMLEngine deklarierte Funktion getLastErrorMessage() verwenden, um zusätzliche Informationen abzurufen.
Rückgabewert:
String, der das serialisierte Ergebnis enthält. Bei einem Fehler wird der leere String zurückgegeben.

enableJavaExtensions
`public void` **enableJavaExtensions**`(boolean bEnable)`
Aktiviert/Deaktiviert die Java-Erweiterungsfunktionen.

enableDotNetExtensions
`public void` **enableDotNetExtensions**`(boolean bEnable)`
Aktiviert/Deaktiviert die .NET-Erweiterungsfunktionen.

IReleasable

Öffentliche Schnittstelle für Release-Funktionalität. Wenn ein Objekt, das diese Schnittstelle implementiert, nicht mehr verwendet wird, muss die Funktion `releaseInstance()` aufgerufen werden, um die Verbindung mit dem COM Server zu trennen. Sobald alle

 © 2012 Altova GmbH

Verbindungen zum COM-Server getrennt sind, wird der COM-Server automatisch heruntergefahren.

Subschnittstelle: IXQuery, IXSLT
Implementierende Klassen: XQuery, XSLT1, XSLT2

Methoden
Es sind die folgenden Methoden definiert.

releaseInstance
`public void releaseInstance()`
Trennt die Verbindung des Objekts zum COM-Server.

IXMLValidator

Schnittstelle für den XML Validator. Öffentliche Schnittstelle, die IAltovaXMLEngine erweitert.

Superschnittstelle: IAltovaXMLEngine, IReleasable
Implementierende Klassen: XMLValidator

Methoden
Es sind die folgenden Methoden definiert.

isValid
`public boolean isValid()`
Validiert die XMl-Input-Daten gegen die/das darin definierte DTD/Schema.
Rückgabewert:
`true` bei erfolgreicher Validierung, `false` bei fehlgeschlagener Validierung. Wenn die Validierung fehlschlägt, können Sie mit Hilfe der in IAltovaXMLEngine deklarierten Funktion getLastErrorMessage() zusätzliche Informationen abrufen.

isWellFormed
`public boolean isWellFormed()`
Überprüft die XML-Input-Daten auf Wohlgeformtheit.
Rückgabewert:
`true` bei Erfolg, `false` bei Fehler. Wenn bei der Überprüfung ein Fehler gefunden wird, können Sie mit Hilfe der in IAltovaXMLEngine deklarierten Funktion getLastErrorMessage() zusätzliche Informationen abrufen.

isValidWithExternalSchemaOrDTD
`public boolean isValidWithExternalSchemaOrDTD()`
Validiert die XML-Input-Daten gegen die externe DTD/das externe Schema, die/das mit den Funktionen `setDTDFileName()`, `setDTDFromText()`, `setSchemaFileName()`, `setSchemaFromText()` definiert werden kann.
Rückgabewert:
`true` bei Erfolg, `false` bei Fehler. Wenn bei der Überprüfung ein Fehler gefunden wird, können Sie mit Hilfe der in IAltovaXMLEngine deklarierten Funktion getLastErrorMessage() zusätzliche Informationen abrufen.

setSchemaFileName
`public void` **`setSchemaFileName`**`(java.lang.String filename)`
Definiert den Dateinamen für das externe Schema.
Parameter:
`filename`: eine absolute URL, die den Basispfad für das Schema angibt.

setDTDFileName
`public void` **`setDTDFileName`**`(java.lang.String filename)`
Definiert den Dateinamen für die externe DTD.
Parameter:
`filename`: eine absolute URL, die den Basispfad für die DTD angibt.

setSchemaFromText
`public void` **`setSchemaFromText`**`(java.lang.String text)`
Definiert den Textwert für das externe Schema.
Parameter:
text: ein String, der das Schema als Text enthält

setDTDFromText
`public void` **`setDTDFromText`**`(java.lang.String text)`
Definiert den Textwert für die externe DTD.
Parameter:
`text`: String, der die DTD als Text enthält.

TreatXBRLInconsistenciesAsErrors
`public void` **`TreatXBRLInconsistenciesAsErrors`**`(boolean bEnable)`
Gibt semantische XBRL-Inkonsistenzen als Fehler zurück, wenn auf `True` gesetzt. Die Standardeinstellung ist `False`.
Parameter:
`bEnable`: boolean

IXQuery

Schnittstelle für den XQuery-Prozessor. Öffentliche Schnittstelle, die `IAltovaXMLEngine` und `IExecutable` erweitert.

Superschnittstelle: `IAltovaXMLEngine`, `IExecutable`, `IReleasable`
Implementierende Klassen: `XQuery`

Methoden
Es sind die folgenden Methoden definiert.

setXQueryFileName
`public void` **`setXQueryFileName`**`(java.lang.String filename)`
Definiert den Dateinamen des XQuery-Dokuments.
Parameter:
`filename`: eine absolute URL, die den Basispfad zur XQuery-Datei angibt.

© 2012 Altova GmbH

setXQueryStatement
`public void` **`setXQueryStatement`**`(java.lang.String text)`
Definiert den Textwert der XQuery-Anweisung.
Parameter:
`text`: ein String, der die XQuery-Anweisung enthält.

setOutputEncoding
`public void` **`setOutputEncoding`**`(java.lang.String encoding)`
Definiert die Kodierung des Ergebnisdokuments.
Parameter:
`encoding`: ein String, der den Namen der Kodierung enthält (z.B.: `UTF-8`, `UTF-16`, `ASCII`, `8859-1`, `1252`).

getOutputEncoding
`public java.lang.String` **`getOutputEncoding`**`()`
Ruft die für das Ergebnisdokument angegebene Kodierung ab.
Rückgabewert:
ein String, der einen Kodierungsnamen enthält.

setOutputIndent
`public void` **`setOutputIndent`**`(boolean indent)`
Aktiviert/deaktiviert die Option zum Einrücken des Texts im Ergebnisdokument.
Parameter:
`indent`: Boolscher Wert zum Aktivieren/Deaktivieren der Einrückung in der Ausgabe.

getOutputIndent
`public boolean` **`getOutputIndent`**`()`
Ruft ab, ob die Ausgabe für das Ergebnisdokument Einrückungen enthalten soll oder nicht.
Rückgabewert:
Boolescher Wert, der angibt, ob die Ausgabe eingerückt werden soll (`true`) oder nicht (`false`).

setOutputMethod
`public void` **`setOutputMethod`**`(java.lang.String method)`
Definiert die Serialisierungsmethode für das Ergebnisdokument.
Parameter:
`method`: ein String, der die Serialisierungsmethode enthält. (Gültige Werte sind: `xml`, `xhtml`, `html`, `text`).

getOutputMethod
`public java.lang.String` **`getOutputMethod`**`()`
Ruft die Serialisierungsmethode für das Ergebnisdokument ab.
Rückgabewert:
ein String, der die Serialisierungsmethode für das Ausgabedokument enthält.

setOutputOmitXMLDeclaration
`public void` **`setOutputOmitXMLDeclaration`**`(boolean decl)`
Aktiviert/deaktiviert die Serialisierungsoption `omitXMLDeclaration` für das Ergebnisdokument.

Parameter:
`decl`: neuer Boolescher Wert für den Parameter `omit-xml-declaration`.

getOutputOmitXMLDeclaration
`public boolean` **`getOutputOmitXMLDeclaration`**`()`
Ruft den Wert der Option omitXMLDeclaration ab, der für das Ergebnisdokument definiert wurde.
Rückgabewert:
Boolescher Wert, der angibt, ob das Ergebnisdokument eine XML-Deklaration enthält (`true`) oder nicht (`false`).

addExternalVariable
```
public void addExternalVariable(java.lang.String name,
                                 java.lang.String val)
```
Fügt einen Namen und Wert für eine externe Variable hinzu.
Parameter:
`name`: ein String, der einen gültigen QName als Variablennamen enthält.
`val`: ein String, der den Wert der Variable enthält; der Wert wird als String verwendet.

addExternalVariableAsXPath
```
public void addExternalVariableAsXPath(java.lang.String name,
                                       java.lang.String val)
```
Fügt einen Namen und Wert für eine externe Variable hinzu, wobei der Wert als XPath 2.0-Ausdruck ausgewertet wird.
Parameter:
`name`: ein String, der einen gültigen QName als Variablennamen enthält..
`val`: ein String, der den Wert der Variable enthält; der Wert wird als XPath 2.0-Ausdruck ausgewertet.

clearExternalVariableList
`public void` **`clearExternalVariableList`**`()`
Löscht die Liste der externen Variablen.

IXSLT

Schnittstelle für XSLT-Prozessoren. Öffentliche Schnittstelle, die `IAltovaXMLEngine` und `IExecutable` erweitert.

Superschnittstelle: `IAltovaXMLEngine`, `IExecutable`, `IReleasable`
Implementierende Klassen: `XSLT1` und `XSLT2`

Hinweis:
Der XSLT 2.0-Prozessor kann zur Verarbeitung eines XSLT 1.0 Stylesheet im Rückwärtskompatibilitätsmodus verwendet werden. Die Ausgabe kann sich allerdings von der eines XSLT 1.0-Prozessors, der dasselbe XSLT 1.0 Stylesheet verarbeitet, unterscheiden.

Methoden
Es sind die folgenden Methoden definiert.

setXSLTFileName
`public void` **`setXSLTFileName`**`(java.lang.String name)`
Definiert den Dateinamen für die XSLT-Daten.
Parameter:

 © 2012 Altova GmbH

`name`: eine absolute URL, die den Basispfad zur XSLT-Datendatei angibt.

setXSLTFromText
`public void setXSLTFromText(java.lang.String text)`
Definiert den Textwert für die XSLT-Daten.
Parameter:
`text`: ein String, der serialisierte XSLT-Daten enthält.

addExternalParameter
```
public void addExternalParameter(java.lang.String name,
                                 java.lang.String val)
```
Fügt den Namen und Wert eines externen Parameters hinzu.
Parameter:
`name`: ein String, der einen gültigen QName als Parameternamen hat.
`val`: ein String, der den Wert des Parameters enthält. Der Wert wird als ein XPath-Ausdruck ausgewertet.

clearExternalParameterList
`public void clearExternalParameterList()`
Löscht die Liste der externen Parameter.

setXSLTStackSize
`public void setXSLTStackSize(long nVal)`
Die Stack-Größe definiert, bis zu welcher Tiefe Anweisungen ausgeführt werden sollen. Wird die Stack-Größe während einer Transformation überschritten, wird ein Fehler zurückgegeben.
Parameter:
`nVal`: numerischer Wert für die neue Stack-Größe. Muss größer als 100 sein. Der anfangs eingestellte Wert ist 1000.

Klassen

Im Folgenden sehen Sie eine Übersicht über die Klassen von `com.altova.engines`. Ausführliche Beschreibungen dazu finden Sie in den entsprechenden Abschnitten.

- `AltovaXMLFactory`
  Erstellt über einen nativen Aufruf eine neue AltovaXML COM Serverobjekt-Instanz und bietet Zugriff auf die AltovaXML Prozessoren.
- `XMLValidator`
  Klasse, die den XMLValidator enthält.
- `XQuery`
  Klasse, die den XQuery 1.0-Prozessor enthält.
- `XSLT1`
  Klasse, die den XSLT 1.0-Prozessor enthält.
- `XSLT2`
  Klasse, die den XSLT 2.0-Prozessor enthält.

AltovaXMLFactory

```
Öffentliche Klasse AltovaXMLFactory
erweitert java.lang.Object
```

implementiert IAltovaXMLFactory

Implementierte Schnittstellen: IAltovaXMLFactory, IReleasable

Beschreibung
Erstellt mittels native Call ein neues AltovaXML COM Serverobjekt und bietet Zugriff auf die AltovaXML-Prozessoren. Die Beziehung zwischen AltovaXMLFactory und dem AltovaXML COM-Objekt ist eine 1:1-Beziehung. Das bedeutet, dass anschließende Aufrufe der getENGINENAMEInstance() Funktion Schnittstellen für dieselbe Prozessorinstanz zurückgeben.

Methoden
Es sind die folgenden Methoden definiert.

getInstance
public static IAltovaXMLFactory **getInstance**()
Erstellt ein neues AltovaXMLFactory Objekt und verbindet es mit einem neuen AltovaXML COM Serverobjekt.
Rückgabewert:
die Schnittstelle IAltovaXMLFactory für das neu erstellte AltovaXMLFactory Objekt oder Null, wenn die Erstellung des COM-Objekts fehlgeschlagen ist. Im zweiteren Fall sollten Sie sicher stellen, dass AltovaXML.exe ordnungsgemäß als COM-Serverobjekt registriert ist.

releaseInstance
public void **releaseInstance**()
Gibt die Verbindung des Objekts mit dem COM-Server frei.
Wird definiert durch:
releaseInstance in der Schnittstelle IReleasable.

getXQueryInstance
public IXQuery **getXQueryInstance**()
Erstellt eine neue Instanz einer XQuery-Klasse für die aktuelle XQuery-Prozessorinstanz. Nach Verwendung muss die Verbindung des Objekts mit dem Prozessor wieder gelöst werden. Verwenden Sie dazu die in der IReleasable Schnittstelle deklarierte Funktion releaseInstance().
Wird definiert durch:
getXQueryInstance in der Schnittstelle IAltovaXMLFactory.
Rückgabewert:
die IXQuery Schnittstelle der neu erstellten Klasse.

getXSLT1Instance
public IXSLT **getXSLT1Instance**()
Erstellt eine neue Instanz der XSLT1 Klasse für die aktuelle XSLT 1.0 Prozessorinstanz. Nach Verwendung muss die Verbindung des Objekts mit dem Prozessor wieder gelöst werden. Verwenden Sie dazu die in der IReleasable Schnittstelle deklarierte Funktion releaseInstance().
Wird definiert durch:
getXSLT1Instance in den Schnittstelle IAltovaXMLFactory.
Rückgabewert:
die IXSLT Schnittstelle der neu erstellten Klasse.

 © 2012 Altova GmbH

getXSLT2Instance
public IXSLT **getXSLT2Instance**()
Erstellt eine neue Instanz der XSLT2 Klasse für die aktuelle XSLT 2.0 Prozessorinstanz. Nach Verwendung muss die Verbindung des Objekts mit dem Prozessor wieder gelöst werden. Verwenden Sie dazu die in der IReleasable Schnittstelle deklarierte Funktion releaseInstance().
Wird definiert durch:
getXSLT2Instance in der Schnittstelle IAltovaXMLFactory.
Rückgabewert:
die IXSLT Schnittstelle der neu erstellten Klasse.

getXMLValidatorInstance
public IXMLValidator **getXMLValidator**()
Erstellt eine neue Instanz der XMLValidator Klasse für die aktuelle XML Validator-Instanz. Nach Verwendung muss die Verbindung des Objekts mit dem Prozessor wieder gelöst werden. Verwenden Sie dazu die in der IReleasable Schnittstelle deklarierte Funktion releaseInstance().
Wird definiert durch:
getXMLValidatorInstance in der Schnittstelle IAltovaXMLFactory.
Rückgabewert:
die IXMLValidator Schnittstelle der neu erstellten Klasse.

Mit den folgenden Methoden können Kataloge für Dokument-Lookup-Zwecke hinzugefügt werden. Nachdem die Kataloge hinzugefügt wurden, werden sie so lange zu Lookup Zwecken herangezogen, bis der COM Server beendet wird. Hinzugefügte Kataloge können nicht mehr entfernt werden.

app.AddXMLCatalogDefault()
Fügt den Altova-Standardkatalog RootCatalog.xml zu den Katalogen hinzu.

app.AddXMLCatalogFromFile(string catalogfilename)
Fügt den durch catalogfilename identifizierten Katalog zu den Katalogen hinzu.

app.AddXMLCatalogFromText(string catalogtext)
Fügt den Katalog mit dem Inhalt catalogtext zu den Katalogen hinzu.

XMLValidator

```
Öffentliche Klasse XMLValidator
erweitert java.lang.Object
implementiert IXMLValidator
```

Implementierte Schnittstellen: IAltovaXMLEngine, IReleasable, IXMLValidator

Beschreibung
Klasse, die den XMLValidator enthält. Keine direkte Konstruktion/kein direkter Zugriff möglich. Erlaubt durch Aufruf der Funktion getXMLValidatorInstance() auf eine Instanz von IAltovaXMLFactory Zugriff auf die IXMLValidator Schnittstelle.

Konstruktoren
Es ist der folgende Konstruktor definiert.

XMLValidator
`geschützter` **XMLValidator**`(lang nValidatorPtr)`

Methoden
Die folgenden Methoden sind definiert.

releaseInstance
`public void` **releaseInstance**`()`
Gibt die Verbindung des Objekts zum COM Server frei.
Definiert durch:
`releaseInstance` in Schnittstelle `IReleasable`.

setInputXMLFileName
`public void` **setInputXMLFileName**`(java.lang.String str)`
Definiert den Dateinamen für die XML-Eingabedaten. Beachten Sie, dass absolute URLs verwendet werden müssen.
Definiert durch:
`setInputXMLFileName` in Schnittstelle `IAltovaXMLEngine`.
Parameter:
`str`: eine absolute URL, die den Basisordner der XML-Daten angibt.

setInputXMLFromText
`public void` **setInputXMLFromText**`(java.lang.String str)`
Definiert den Textwert für die XML-Eingabedaten. Beispiel: `setInputXMLFromText( "<doc> <a>text</a> </doc>" )`
Definiert durch:
`setInputXMLFromText` in der Schnittstelle `IAltovaXMLEngine`.
Parameter:
`str`: ein String, der XML-Daten enthält.

getLastErrorMessage
`public java.lang.String` **getLastErrorMessage**`()`
Ruft die letzte Fehlermeldung vom Prozessor ab.
Definiert durch:
`getLastErrorMessage` in Schnittstelle `IAltovaXMLEngine`.
Rückgabewert:
ein String, der die letzte Fehlermeldung enthält.

isValid
`public boolean` **isValid**`()`
Validiert die XML-Eingabedaten gegen die darin definierte DTD/das Schema.
Definiert durch:
`isValid` in Schnittstelle `IXMLValidator`.
Rückgabewert:
`true` bei erfolgreicher Validierung, `false` bei fehlgeschlagener Validierung. Bei einer fehlgeschlagenen Validierung können Sie mit Hilfe der in `IAltovaXMLEngine` deklarierten Funktion `getLastErrorMessage` zusätzliche Informationen abrufen.

© 2012 Altova GmbH

isWellFormed
`public boolean` **`isWellFormed`**`()`
Überprüft die XML-Eingabedaten auf Wohlgeformtheit.
Definiert durch:
`isWellFormed` in Schnittstelle `IXMLValidator`.
Rückgabewert:
`true` bei Wohlgeformtheit, `false` bei Fehler. Falls die Wohlgeformtheitsprüfung fehlschlägt, können Sie mit Hilfe der in `IAltovaXMLEngine` deklarierten Funktion `getLastErrorMessage` zusätzliche Informationen abrufen.

isValidWithExternalSchemaOrDTD
`public boolean` **`isValidWithExternalSchemaOrDTD`**`()`
Validiert die XML-Eingabedaten gegen die externe DTD / das externe Schema, welche mit Hilfe der Funktionen `setDTDFileName()`, `setDTDFromText()`, `setSchemaFileName()` und `setSchemaFromText()` definiert werden können. *Eine Beschreibung dieser Funktionen finden Sie weiter unten*.
Definiert durch:
`isValidWithExternalSchemaOrDTD` in Schnittstelle `IXMLValidator`.
Rückgabewert:
`true` bei erfolgreicher Validierung, `false` bei fehlgeschlagener Validierung. Wenn die Validierung fehlschlägt, können Sie mit Hilfe der in `IAltovaXMLEngine` definierten Funktion `getLastErrorMessage` zusätzliche Informationen abrufen.

setSchemaFileName
`public void` **`setSchemaFileName`**`(java.lang.String str)`
Definiert den Dateinamen des externen Schemas.
Definiert durch:
`setSchemaFileName` in Schnittstelle `IXMLValidator`.
Parameter:
`str`: eine absolute URL, die den Basisordner des Schemas angibt.

setDTDFileName
`public void` **`setDTDFileName`**`(java.lang.String str)`
Definiert den Dateinamen der externen DTD.
Definiert durch:
`setDTDFileName` in Schnittstelle `IXMLValidator`.
Parameter:
`str`: eine absolute URL, die den Basisordner der DTD angibt.

setSchemaFromText
`public void` **`setSchemaFromText`**`(java.lang.String str)`
Definiert den Textwert für das externe Schema.
Definiert durch:
setSchemaFromText in Schnittstelle `IXMLValidator`.
Parameter:
`str`: ein String, der Schema als Text enthält.

setDTDFromText
`public void` **`setDTDFromText`**`(java.lang.String str)`

Definiert den Textwert für die externe DTD.
Definiert durch:
setDTDFromText in Schnittstelle IXMLValidator.
Parameter:
str: ein String, der DTD als Text enthält.

XQuery

```
Öffentliche Klasse XQuery
erweitert java.lang.Object
implementiert IXQuery
```

Implementierte Schnittstellen: IAltovaXMLEngine, IExecutable, IReleasable, IXQuery

Beschreibung
Klasse, die den XQuery-Prozessor enthält. Keine direkte Konstruktion/kein direkter Zugriff möglich. Erlaubt durch Aufruf der Funktion getXQueryInstance() auf eine Instanz von IAltovaXMLFactory Zugriff auf die IXQuery Schnittstelle.

Konstruktoren
Es ist der folgende Konstruktor definiert.

XQuery

```
geschützt XQuery(lang nXQueryPtr)
```

Methoden
Die folgenden Methoden sind definiert.

releaseInstance

```
public void releaseInstance()
```

Gibt die Verbindung des Objekts zum COM Server frei.
Definiert durch:
releaseInstance in Schnittstelle IReleasable.

execute

```
public boolean execute(java.lang.String sOutFile)
```

Führt die Abfrage aus und speichert das Ergebnis in einer Datei. Bei einem Fehler können Sie mit Hilfe der in IAltovaXMLEngine deklarierten Funktion getLastErrorMessage() zusätzliche Informationen abrufen.
Definiert durch:
execute in Schnittstelle IExecutable.
Parameter:
sOutFile: eine absolute URL, die den Ordner der Ausgabedatei angibt.
Rückgabewert:
true bei erfolgreicher Ausführung, false bei Fehler

executeAndGetResultAsString

```
public java.lang.String executeAndGetResultAsString()
```

Führt die Abfrage durch und gibt das Ergebnis in Form eines UTF 16-Textstring zurück. Bei einem Fehler können Sie mit Hilfe der in IAltovaXMLEngine deklarierten Funktion getLastErrorMessage() zusätzliche Informationen abrufen.

 © 2012 Altova GmbH

Definiert durch:
executeAndGetResultAsString in Schnittstelle IExecutable.
Rückgabewert:
String, der das serialisierte Ergebnis enthält. Bei einem Fehler wird der leere String zurückgegeben.

setInputXMLFileName
`public void setInputXMLFileName(java.lang.String str)`
Definiert den Dateinamen für die XML-Eingabedaten. Beachten Sie, dass absolute URLs verwendet werden müssen.
Definiert durch:
setInputXMLFileName in Schnittstelle IAltovaXMLEngine.
Parameter:
`str`: eine absolute URL, die den Basisordner der XML-Daten angibt.

setInputXMLFromText
`public void setInputXMLFromText(java.lang.String str)`
Definiert den Textwert für die XML-Eingabedaten. Beispiel: `setInputXMLFromText( "<doc> <a>text</a> </doc>" )`
Definiert durch:
setInputXMLFromText in der Schnittstelle IAltovaXMLEngine.
Parameter:
`str`: ein String, der XML-Daten enthält.

getLastErrorMessage
`public java.lang.String getLastErrorMessage()`
Ruft die letzte Fehlermeldung vom Prozessor ab.
Definiert durch:
getLastErrorMessage in Schnittstelle IAltovaXMLEngine.
Rückgabewert:
ein String, der die letzte Fehlermeldung enthält.

setXQueryFileName
`public void setXQueryFileName(java.lang.String str)`
Definiert den Dateinamen des XQuery-Dokuments.
Definiert durch:
setXQueryFileName in Schnittstelle IXQuery.
Parameter:
`str`:eine absolute URL, die den Basisordner der XQuery-Datei angibt.

setXQueryStatement
`public void setXQueryStatement(java.lang.String str)`
Definiert den Textwert für die XQuery-Anweisung
Definiert durch:
setXQueryStatement in Schnittstelle IXMLValidator.
Parameter:
`str`: ein String, der die XQuery-Anweisung als Text enthält.

setOutputEncoding
`public void setOutputEncoding(java.lang.String str)`
Definiert die Kodierung für das Ergebnisdokument.

Definiert durch:
setOutputEncoding in Schnittstelle IXQuery.
Parameter:
str: ein String der den Namen der Kodierung enthält (z.B.: UTF-8, UTF-16, ASCII, 8859-1, 1252)

getOutputEncoding
public java.lang.String **getOutputEncoding**()
Ruft die für das Ergebnisdokument definierte Kodierung auf.
Definiert durch:
getOutputEncoding in Schnittstelle IXQuery.
Rückgabewert:
ein String, der den Namen der Kodierung enthält.

setOutputIndent
public void **setOutputIndent**(boolean bVal)
Aktiviert/deaktiviert die Einrückoption für das Ergebnisdokument.
Definiert durch:
setOutputIndent in Schnittstelle IXQuery.
Parameter:
bVal: Boolescher Wert zum Aktivieren/Deaktivieren der Einrückung.

getOutputIndent
public boolean **getOutputIndent**()
Ruft die für das Ergebnisdokument definierte Einrückoption für das Ausgabedokument ab.
Definiert durch:
getOutputIndent in Schnittstelle IXQuery.
Rückgabewert:
der aktuelle Wert des Serialisierungsparameters für die Einrückung.

setOutputMethod
public void **setOutputMethod**(java.lang.String str)
Definiert die Serialisierungsmethode für das Ergebnisdokument.
Definiert durch:
setOutputMethod in Schnittstelle IXQuery.
Parameter:
str: ein String, der die Serialisierungsmethode enthält. Gültige Werte: xml, xhtml, html, text.

getOutputMethod
public java.lang.String **getOutputMethod**()
Ruft die Serialisierungsmethode für das Ergebnisdokument ab.
Definiert durch:
getOutputMethod in Schnittstelle IXQuery.
Rückgabewert:
die aktuelle Serialisierungsmethode.

setOutputOmitXMLDeclaration

 © 2012 Altova GmbH

```
public void setOutputOmitXMLDeclaration(boolean bVal)
```
Aktiviert/deaktiviert die Serialisierungsoption `omitXMLDeclaration` für das Ergebnisdokument.
Definiert durch:
`setOutputOmitXMLDeclaration` in Schnittstelle `IXQuery`.
Parameter:
`bVal`: ein neuer Boolescher Wert für den `omit-xml-declaration` Parameter.

getOutputOmitXMLDeclaration
```
public boolean getOutputOmitXMLDeclaration()
```
Ruft den Wert der Option `omitXMLDeclaration,` die für das Ergebnisdokument definiert wurde, ab.
Definiert durch:
`getOutputOmitXMLDeclaration` in Schnittstelle `IXQuery`.
Rückgabewert:
Boolescher Wert des `omit-xml-declaration` Parameters.

addExternalVariable
```
public void addExternalVariable(java.lang.String strName,
                                java.lang.String strVal)
```
Fügt den Namen und Wert einer externen Variable hinzu.
Definiert durch:
`addExternalVariable` in Schnittstelle `IXQuery`.
Parameter:
`strName`: ein String, der einen gültigen QNamen als Variablennamen, enthält.
`strVal`: ein String, der den Wert der Variablen enthält; dieser Wert wird als String verwendet.

addExternalVariableAsXPath
```
public void addExternalVariableAsXPath(java.lang.String strName,
                                       java.lang.String strVal)
```
Fügt den Namen und Wert für eine externe Variable hinzu, wobei der Wert als XPath 2.0-Ausdruck ausgewertet wird.

Definiert durch:
`addExternalVariableAsXPath` in Schnittstelle `IXQuery`.
Parameter:
`strName`: ein String, der einen gültigen QName als Variablennamen enthält.
`strVal`: ein String, der den Wert der Variablen enthält; dieser Wert wird als XPath 2.0-Ausdruck ausgewertet.

clearExternalVariableList
```
public void clearExternalVariableList()
```
Löscht die Liste der externen Variablen.
Definiert durch:
`clearExternalVariableList` in Schnittstelle `IXQuery`.

enableJavaExtensions
```
public void enableJavaExtensions(boolean bEnable)
```
Aktiviert/Deaktiviert die Java-Erweiterungsfunktionen.
Wird definiert durch:
`enableJavaExtensions` in der Schnittstelle `IExecutable`.

enableDotNetExtensions
`public void` **`enableDotNetExtensions`**`(boolean bEnable)`
Aktiviert/Deaktiviert die .NET-Erweiterungsfunktionen.
Wird definiert durch:
`enableJavaExtensions` in der Schnittstelle `IExecutable`.

XSLT1

```
Öffentliche Klasse XSLT1
erweitert java.lang.Object
implementiert IXSLT
```

Implementierte Schnittstellen: `IAltovaXMLEngine`, `IExecutable`, `IReleasable`, `IXSLT`

Beschreibung
Klasse, die den XSLT 1.0-Prozessor enthält. Keine direkte Konstruktion/kein direkter Zugriff möglich. Erlaubt durch Aufruf der Funktion `getXSLT1Instance()` auf eine Instanz von `IAltovaXMLFactory` Zugriff auf die `IXSLT` Schnittstelle.

Konstruktoren
Es ist der folgende Konstruktor definiert.

XSLT1
`geschützt` **`XSLT1`**`(lang nXSLT1Ptr)`

Methoden
Die folgenden Methoden sind definiert.

releaseInstance
`public void` **`releaseInstance`**`()`
Gibt die Verbindung des Objekts zum COM Server frei.
Definiert durch:
`releaseInstance` in Schnittstelle `IReleasable`.

execute
`public boolean` **`execute`**`(java.lang.String sOutFile)`
Führt die Abfrage aus und speichert das Ergebnis in einer Datei. Bei einem Fehler können Sie mit Hilfe der in `IAltovaXMLEngine` deklarierten Funktion `getLastErrorMessage()` zusätzliche Informationen abrufen.
Definiert durch:
`execute` in Schnittstelle `IExecutable`.
Parameter:
`sOutFile`: eine absolute URL, die den Ordner der Ausgabedatei angibt.
Rückgabewert:
`true` bei erfolgreicher Ausführung, `false` bei Fehler

executeAndGetResultAsString
`public java.lang.String` **`executeAndGetResultAsString`**`()`
Führt die Anweisung durch und gibt das Ergebnis in Form eines UTF 16-Textstring zurück. Bei einem Fehler können Sie mit Hilfe der in `IAltovaXMLEngine` deklarierten Funktion

 © 2012 Altova GmbH

getLastErrorMessage() zusätzliche Informationen abrufen.
Definiert durch:
executeAndGetResultAsString in Schnittstelle IExecutable.
Rückgabewert:
String, der das serialisierte Ergebnis enthält. Bei einem Fehler wird der leere String zurückgegeben.

setInputXMLFileName
`public void setInputXMLFileName(java.lang.String str)`
Definiert den Dateinamen für die XML-Eingabedaten. Beachten Sie, dass absolute URLs verwendet werden müssen.
Definiert durch:
setInputXMLFileName in Schnittstelle IAltovaXMLEngine.
Parameter:
`str`: eine absolute URL, die den Basisordner der XML-Daten angibt.

setInputXMLFromText
`public void setInputXMLFromText(java.lang.String str)`
Definiert den Textwert für die XML-Eingabedaten. Beispiel: `setInputXMLFromText( "<doc> <a>text</a> </doc>" )`
Definiert durch:
setInputXMLFromText in der Schnittstelle IAltovaXMLEngine.
Parameter:
`str`: ein String, der XML-Daten enthält.

getLastErrorMessage
`public java.lang.String getLastErrorMessage()`
Ruft die letzte Fehlermeldung vom Prozessor ab.
Definiert durch:
getLastErrorMessage in Schnittstelle IAltovaXMLEngine.
Rückgabewert:
ein String, der die letzte Fehlermeldung enthält.

setXSLTFileName
`public void setXSLTFileName(java.lang.String str)`
Definiert den Dateinamen für die XSLT-Daten.
Definiert durch:
setXSLTFileName in Schnittstelle IXSLT.
Parameter:
`str`: eine absolute URL, die den Basisordner der XSLT-Daten angibt.

setXSLTFromText
`public void setXSLTFromText(java.lang.String str)`
Definiert den Textwert für die XSLT-Daten.
Definiert durch:
setXSLTFromText in Schnittstelle IXSLT.
Parameter:
`str`: ein String, der die serialisierten XSLT-Daten enthält.

addExternalParameter

```
public void addExternalParameter(java.lang.String strName,
                                 java.lang.String strVal)
```
Fügt den Namen und Wert eines externen Parameters hinzu.
Definiert durch:
addExternalParameter in Schnittstelle IXSLT.
Parameter:
strName: ein String, der einen gültigen QNamen als Parameternamen, enthält.
strVal: ein String, der den Wert des Parameters enthält; dieser Wert wird als XPath-Ausdruck ausgewertet.

clearExternalParameterList
```
public void clearExternalParameterList()
```
Löscht die Liste der externen Parameter.
Definiert durch:
clearExternalParameterList in Schnittstelle IXSLT.

setXSLTStackSize
```
public void setXSLTStackSize(long nVal)
```
Die Stack Size (Stapelgröße) gibt die maximale Tiefe für die ausgeführten Anweisungen an. Wenn die Stapelgröße bei der Transformation überschritten wird, wird ein Fehler ausgegeben.
Definiert durch:
setXSLTStackSize in Schnittstelle IXSLT.
Parameter:
nVal: numerischer Wert für neue Stapelgröße. Muss größer als 100 sein. Der Anfangswert ist 1000.

enableJavaExtensions
```
public void enableJavaExtensions(boolean bEnable)
```
Aktiviert/Deaktiviert die Java-Erweiterungsfunktionen.
Wird definiert durch:
enableJavaExtensions in der Schnittstelle IExecutable.

enableDotNetExtensions
```
public void enableDotNetExtensions(boolean bEnable)
```
Aktiviert/Deaktiviert die .NET-Erweiterungsfunktionen.
Wird definiert durch:
enableJavaExtensions in der Schnittstelle IExecutable.

XSLT2

```
Öffentliche Klasse XSLT2
erweitert java.lang.Object
implementiert IXSLT
```

Implementierte Schnittstellen: IAltovaXMLEngine, IExecutable, IReleasable, IXSLT

Beschreibung
Klasse, die den XSLT 2.0-Prozessor enthält. Keine direkte Konstruktion/kein direkter Zugriff möglich. Erlaubt durch Aufruf der Funktion getXSLT2Instance() auf eine Instanz von IAltovaXMLFactory Zugriff auf die IXSLT Schnittstelle. Beachten Sie: Der XSLT 2.0-Prozessor kann zur Verarbeitung eines XSLT 1.0 Stylesheet im Rückwärtskompatibilitätsmodus verwendet werden. Die Ausgabe kann sich allerdings von der eines XSLT 1.0-Prozessors, der dasselbe XSLT 1.0 Stylesheet verarbeitet, unterscheiden.

© 2012 Altova GmbH

Konstruktoren
Es ist der folgende Konstruktor definiert.

XSLT2
`geschützt XSLT2(lang nXSLT2Ptr)`

Methoden
Die folgenden Methoden sind definiert.

releaseInstance
`public void releaseInstance()`
Gibt die Verbindung des Objekts zum COM Server frei.
Definiert durch:
`releaseInstance` in Schnittstelle `IReleasable`.

execute
`public boolean execute(java.lang.String sOutFile)`
Führt die Abfrage aus und speichert das Ergebnis in einer Datei. Bei einem Fehler können Sie mit Hilfe der in `IAltovaXMLEngine` deklarierten Funktion `getLastErrorMessage()` zusätzliche Informationen abrufen.
Definiert durch:
`execute` in Schnittstelle `IExecutable`.
Parameter:
`sOutFile`: eine absolute URL, die den Ordner der Ausgabedatei angibt.
Rückgabewert:
`true` bei erfolgreicher Ausführung, `false` bei Fehler

executeAndGetResultAsString
`public java.lang.String executeAndGetResultAsString()`
Führt die Anweisung durch und gibt das Ergebnis in Form eines UTF-16-Textstring zurück. Bei einem Fehler können Sie mit Hilfe der in `IAltovaXMLEngine` deklarierten Funktion `getLastErrorMessage()` zusätzliche Informationen abrufen.
Definiert durch:
`executeAndGetResultAsString` in Schnittstelle `IExecutable`.
Rückgabewert:
String, der das serialisierte Ergebnis enthält. Bei einem Fehler wird der leere String zurückgegeben.

setInputXMLFileName
`public void setInputXMLFileName(java.lang.String str)`
Definiert den Dateinamen für die XML-Eingabedaten. Beachten Sie, dass absolute URLs verwendet werden müssen.
Definiert durch:
`setInputXMLFileName` in Schnittstelle `IAltovaXMLEngine`.
Parameter:
`str`: eine absolute URL, die den Basisordner der XML-Daten angibt.

setInputXMLFromText

```
public void setInputXMLFromText(java.lang.String str)
```
Definiert den Textwert für die XML-Eingabedaten. Beispiel: `setInputXMLFromText( "<doc> <a>text</a> </doc>" )`
Definiert durch:
`setInputXMLFromText` in der Schnittstelle `IAltovaXMLEngine`.
Parameter:
`str`: ein String, der XML-Daten enthält.

getLastErrorMessage
```
public java.lang.String getLastErrorMessage()
```
Ruft die letzte Fehlermeldung vom Prozessor ab.
Definiert durch:
`getLastErrorMessage` in Schnittstelle `IAltovaXMLEngine`.
Rückgabewert:
ein String, der die letzte Fehlermeldung enthält.

setXSLTFileName
```
public void setXSLTFileName(java.lang.String str)
```
Definiert den Dateinamen für die XSLT-Daten.
Definiert durch:
`setXSLTFileName` in Schnittstelle `IXSLT`.
Parameter:
`str`: eine absolute URL, die den Basisordner der XSLT-Daten angibt.

setXSLTFromText
```
public void setXSLTFromText(java.lang.String str)
```
Definiert den Textwert für die XSLT-Daten.
Definiert durch:
`setXSLTFromText` in Schnittstelle `IXSLT`.
Parameter:
`str`: ein String, der die serialisierten XSLT-Daten enthält.

addExternalParameter
```
public void addExternalParameter(java.lang.String strName,
                                 java.lang.String strVal)
```
Fügt den Namen und Wert eines externen Parameters hinzu.
Definiert durch:
`addExternalParameter` in Schnittstelle `IXSLT`.
Parameter:
`strName`: ein String, der einen gültigen QName als Parameternamen, enthält.
`strVal`: ein String, der den Wert des Parameters enthält; dieser Wert wird als XPath-Ausdruck ausgewertet.

clearExternalParameterList
```
public void clearExternalParameterList()
```
Löscht die Liste der externen Parameter.
Definiert durch:
`clearExternalParameterList` in Schnittstelle `IXSLT`.

setInitialTemplateName
```
public void setInitialTemplateName(java.lang.String str)
```
Definiert den anfangs verwendeten Vorlagennamen für die Transformation.

 © 2012 Altova GmbH

setInitialTemplateMode
`public void` **`setInitialTemplateMode`**`(java.lang.String str)`
Definiert den anfangs verwendeten Vorlagenmodus für die Transformation.

setXSLTStackSize
`public void` **`setXSLTStackSize`**`(long nVal)`
Die Stack Size (Stapelgröße) gibt die maximale Tiefe für die ausgeführten Anweisungen an. Wenn die Stapelgröße bei der Transformation überschritten wird, wird ein Fehler ausgegeben.
Definiert durch:
`setXSLTStackSize` in Schnittstelle `IXSLT`.
Parameter:
`nVal`: numerischer Wert für neue Stapelgröße. Muss größer als 100 sein. Der Anfangswert ist 1000.

enableJavaExtensions
`public void` **`enableJavaExtensions`**`(boolean bEnable)`
Aktiviert/Deaktiviert die Java-Erweiterungsfunktionen.
Wird definiert durch:
`enableJavaExtensions` in der Schnittstelle `IExecutable`.

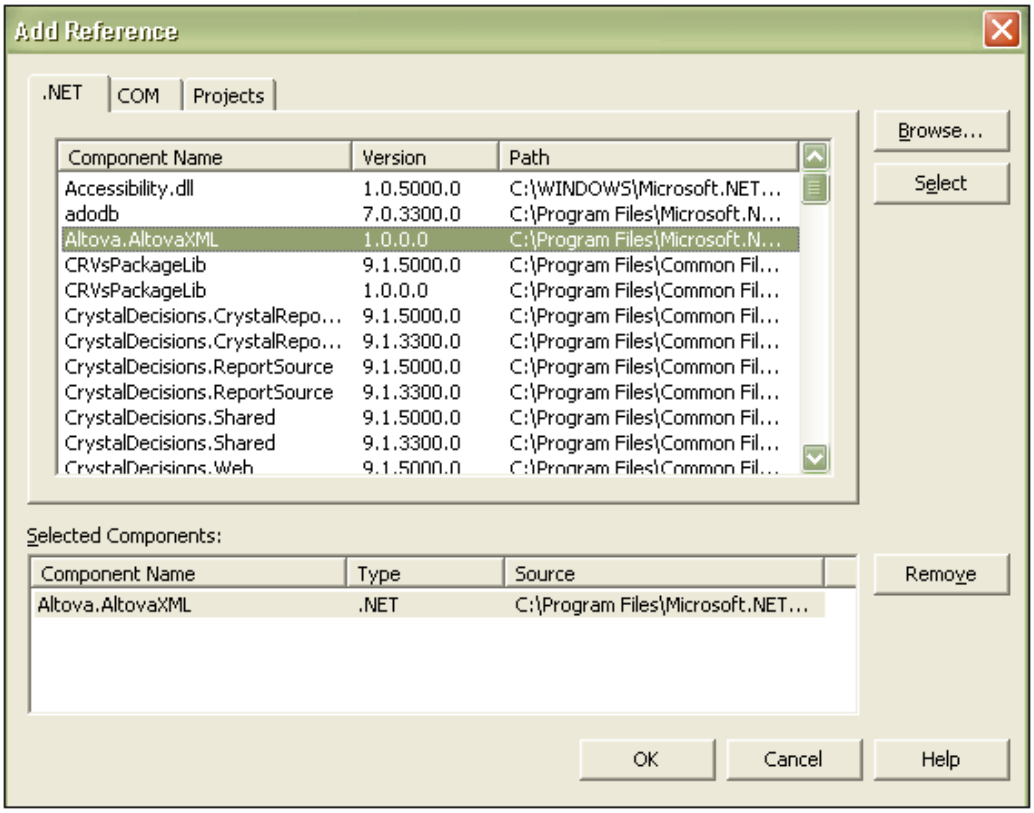
enableDotNetExtensions
`public void` **`enableDotNetExtensions`**`(boolean bEnable)`
Aktiviert/Deaktiviert die .NET-Erweiterungsfunktionen.
Wird definiert durch:
`enableJavaExtensions` in der Schnittstelle `IExecutable`.

2.4 .NET-Schnittstelle

Die .NET-Schnittstelle ist als Hülle rund um die AltovaXML COM-Schnittstelle gebaut. Sie dient als primäre von Altova signierte Interop-Assembly und verwendet den Namespace `Altova.AltovaXML`. Um AltovaXML in Ihrem .NET-Projekt verwenden zu können, müssen Sie: (i) eine Referenz zur AltovaXML DLL (welche den Namen `Altova.AltovaXML.dll` trägt) in Ihrem Projekt hinzufügen und (ii) AltovaXML als COM Serverobjekt registriert haben. Sobald diese (unten beschriebenen) Anforderungen erfüllt sind, können Sie die AltovaXML Funktionalitäten in Ihrem Projekt nutzen.

Hinzufügen der AltovaXML DLL als Referenz zum Projekt
Das AltovaXML Paket enthält eine signierte DLL-Datei namens `Altova.AltovaXML.dll`, die bei der Installation von AltovaXML mit Hilfe des AltovaXML Installationsprogramms automatisch zum globalen Assembly Cache (und zur .NET Referenzbibliothek) hinzugefügt wird. (Normalerweise im Ordner `C:\WINDOWS\assembly`) . Um diese DLL als Referenz in einem .NET-Projekt hinzuzufügen, gehen Sie folgendermaßen vor:

1. Klicken Sie bei geöffnetem .NET-Projekt auf **Project | Add Reference**. Daraufhin wird das Dialogfeld "Add Reference" (*Abbildung unten*) angezeigt und Sie sehen darin eine Liste der installierten .NET-Komponenten. (Anmerkung: Wenn sich die AltovaXML-Komponente nicht in der Liste auf dem Register .NET befindet, kann sie vom Register COM ausgewählt werden.)



2. Wählen Sie in der Komponentenliste `Altova.AltovaXML` aus, doppelklicken Sie darauf oder klicken Sie auf die Schaltfläche "Select" und anschließend auf OK.

Registrieren von AltovaXML als COM-Serverobjekt
Die COM-Registierung erfolgt automatisch durch das AltovaXML Installationsprogramm. Wenn

© 2012 Altova GmbH

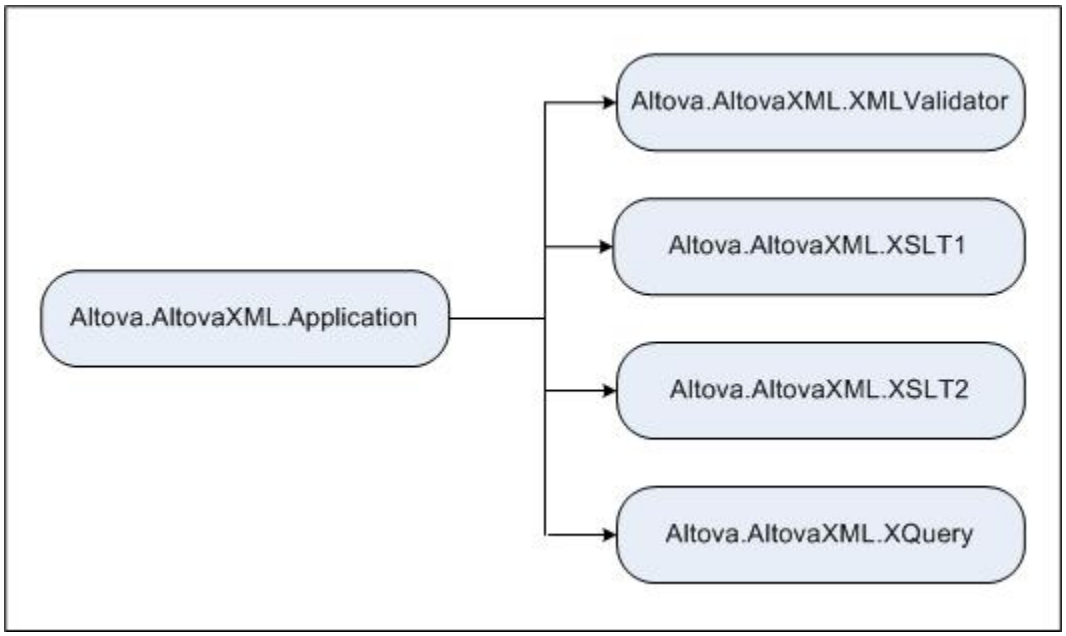
Sie den Ordner der Datei `AltovaXML_COM.exe` nach der Installation ändern, sollten Sie AltovaXML als COM-Serverobjekt registrieren. Führen Sie dazu den Befehl `AltovaXML_COM.exe /regserver` aus. (Beachten Sie, dass der richtige Pfad zur Datei `AltovaXML_COM.exe` eingegeben werden muss. Nähere Informationen siehe Registrieren von AltovaXML als COM-Serverobjekt.)

Sobald die `Altova.AltovaXML.dll` der .NET-Schnittstelle zur Verfügung steht und AltovaXML als COM-Serverobjekt registriert wurde, stehen die AltovaXML Funktionalitäten in Ihrem .NET-Projekt zur Verfügung.

Anmerkung: Falls Sie einen Zugriffsfehler erhalten, vergewissern Sie sich, dass die Berechtigungen richtig eingestellt sind. Gehen Sie zu "Component Services" und geben Sie demselben Benutzerkonto, über das der Application Pool, der AltovaXML enthält, ausgeführt wird, Zugriffsberechtigungen.

2.4.1 Allgemeine Verwendung

Die Klassen und Methoden, die Ihnen zur Verwendung zur Verfügung stehen, sind die im Abschnitt COM-Schnittstelle beschriebenen, befinden sich aber im Namespace `Altova.AltovaXML`. Sie sind in den folgenden Abschnitten aufgelistet. Ausgangspunkt ist das `Altova.AltovaXML.Application` Objekt. Wenn Sie dieses Objekt erstellen, wird eine Verbindung zu einem neuen AltovaXML COM Serverobjekt erstellt. Das Objektmodell wird im Diagramm unten gezeigt.



Methoden
Mit den folgenden Methoden, die im application-Objekt zur Verfügung stehen, können Kataloge für Dokument-Lookup-Zwecke hinzugefügt werden. Nachdem die Kataloge hinzugefügt wurden, werden sie so lange zu Lookup Zwecken herangezogen, bis der COM Server beendet wird. Hinzugefügte Kataloge können nicht mehr entfernt werden.

`app.AddXMLCatalogDefault()`
Fügt den Altova-Standardkatalog `RootCatalog.xml` zu den Katalogen hinzu.

`app.AddXMLCatalogFromFile( string catalogfilename )`
Fügt den durch `catalogfilename` identifizierten Katalog zu den Katalogen hinzu.

`app.AddXMLCatalogFromText( string catalogtext )`
Fügt den Katalog mit dem Inhalt `catalogtext` zu den Katalogen hinzu.

Beispiel
Die Verwendung der AltovaXML Klassen und Methoden im .NET Framework wird unten anhand eines Beispiels im C#-Code für ein Button Event gezeigt. Ein ausführlicheres Beispiel finden Sie am Ende der .NET-Schnittstelle.

```
private void button1_Click(object sender, System.EventArgs e)
{
      Altova.AltovaXML.ApplicationClass appXML = new
Altova.AltovaXML.ApplicationClass();
      Altova.AltovaXML.XMLValidator XMLValidator =
```

 © 2012 Altova GmbH

```
appXML.XMLValidator;
      XMLValidator.InputXMLFromText = "<test>Is this data well-formed?
<a></test>";
      if ( XMLValidator.IsWellFormed() )
      {
      MessageBox.Show( this, "The input data is well-formed") ;
      }
      else
      {
      MessageBox.Show( this, "The input data is not well-formed") ;
      }
}
```

Der oben angeführte Code führt Folgendes aus:

1. Das `Altova.AltovaXML.ApplicationClass` Objekt wird erstellt. Dieses Objekt stellt eine Verbindung zu einem neuen AltovaXML COM Serverobjekt her.
2. Die XML-Validator-Funktionalität wird über `Altova.AltovaXML.XMLValidator.` aufgerufen.
3. Die `InputXMLFromText` Eigenschaft von `Altova.AltovaXML.XMLValidator` liefert die XML-Eingabedaten.
4. Die `IsWellFormed` Methode von `Altova.AltovaXML.XMLValidator` überprüft, ob die vorliegenden XML-Daten wohlgeformt sind und gibt den Wert `TRUE` oder `FALSE` zurück.

Siehe dazu auch die Beispieldateien im Ordner `AltovaXMLExamples` des Applikationsordners.

2.4.2 Altova.AltovaXML.XMLValidator

Beschreibung
Das `Altova.AltovaXML.XMLValidator` Objekt stellt Methoden bereit um Folgendes zu überprüfen:

- die Wohlgeformtheit eines XML-Dokuments.
- die Gültigkeit eines XML-Dokuments gemäß einer DTD oder eines im XML-Dokument referenzierten XML-Schemas.
- die Gültigkeit eines XML-Dokuments gemäß einer DTD oder eines XML-Schemas, die/das extern über den Code bereitgestellt wird.
- die Gültigkeit eines XBRL-Dokuments gemäß einer XBRL-Taxonomie (`.xsd`-Datei)

All diese Methoden geben die Booleschen Werte `TRUE` oder `FALSE` zurück.

Anmerkung: Wenn String-Eingabewerte als URLs interpretiert werden sollen, sollten absolute Pfade verwendet werden. Bei Verwendung eines relativen Pfads sollte im aufrufenden Modul ein Mechanismus zur Auflösung des relativen Pfads definiert sein.

Methoden
Die folgenden Methoden stehen zur Verfügung:

`IsWellFormed()` `as Boolean`
`IsWellFormed` überprüft die Wohlgeformtheit des XML-Dokuments. Gibt den Wert `TRUE` zurück, wenn das XML-Dokument wohlgeformt ist und den Wert `FALSE`, wenn es nicht wohlgeformt ist.

`IsValid()` `as Boolean`
`IsValid` validiert das XML-Dokument gegen die DTD oder das XML-Schema, die/das im XML-Dokument referenziert ist. Gibt den Wert `TRUE` zurück, wenn das XML-Dokument gültig ist und bei Ungültigkeit den Wert `FALSE`. Um ein Dokument gegen eine DTD oder ein XML-Schema zu validieren, die/das nicht im XML-Dokument referenziert ist, verwenden Sie die Methode `IsValidWithExternalSchemaOrDTD`.

`IsValidWithExternalSchemaOrDTD()` `as Boolean`
`IsValidWithExternalSchemaOrDTD` validiert das XML-Dokument gegen eine DTD oder ein XML-Schema, die/das durch eine der folgenden Eigenschaften bereitgestellt wird: `SchemaFileName`, `DTDFileName`, `SchemaFromText` oder `DTDFromText`. Wenn für mehr als eine dieser Eigenschaften Werte definiert sind, verwendet die `IsValidWithExternalSchemaOrDTD` Methode die zuletzt definierte Eigenschaft. Gibt den Wert `TRUE` zurück, wenn das XML-Dokument gültig ist und bei Ungültigkeit den Wert `FALSE`. Um das Dokument gegen eine DTD oder ein XML-Schema zu validieren, die/das im XML-Dokument referenziert ist, verwenden Sie die Methode `IsValid`.

Anmerkung: Die Validierung und Wohlgeformtheitsprüfung muss immer erfolgen, nachdem das XML-Dokument und/oder die DTD bzw. das XML-Schema-Dokument den entsprechenden Eigenschaften zugewiesen wurde.

Eigenschaften
Die folgenden Eigenschaften sind definiert:

`InputXMLFileName`
Ein Eingabestring, der als URL gelesen wird. Definiert den Pfad zu der zu validierenden XML-Datei.

 © 2012 Altova GmbH

SchemaFileName
Ein Eingabestring, der als URL gelesen wird. Definiert den Pfad zur XML-Schemadatei, gegen die das XML-Dokument validiert werden soll.

DTDFileName
Ein Eingabestring, der als URL gelesen wird. Definiert den Pfad zur DTD-Datei, gegen die das XML-Dokument validiert werden soll.

InputXMLFromText
Ein Eingabestring, der ein XML-Dokument erstellt.

SchemaFromText
Ein Eingabestring, der ein XML-Schema-Dokument erstellt.

DTDFromText
Ein Eingabestring, der ein DTD-Dokument erstellt.

LastErrorMessage
Gibt die letzte Fehlermeldung zurück.

TreatXBRLInconsistenciesAsErrors
Gibt semantische XBRL-Inkonsistenzen als Fehler zurück, wenn auf `True` gesetzt. Die Standardeinstellung ist `False`.

Beispiel
Im folgenden C#-Codefragment wird gezeigt, wie man ein XML-Dokument validiert. Ein ausführlicheres Beispiel finden Sie am Ende des Abschnitts .NET-Schnittstelle.

Um diese Codefragmente in einem C#-Projekt zu erstellen, gehen Sie folgendermaßen vor:

1. Fügen Sie in Microsoft Visual Studio mit dem Befehl **Datei | Neu | Projekt** ein neues Projekt hinzu.
2. Fügen Sie durch Klicken auf **Projekt | Referenz hinzufügen** eine Referenz zur AltovaXML DLL hinzu. Daraufhin wird das Dialogfeld "Referenz hinzufügen" angezeigt, in dem eine Liste der installierten .NET-Komponenten angezeigt wird. Wählen Sie darin die AltovaXML-Komponente aus um sie hinzuzufügen. (Anmerkung: Wenn sich die AltovaXML-Komponente nicht in der Liste auf dem Register .NET befindet, kann sie vom Register COM ausgewählt werden.)
3. Geben Sie das unten gezeigte Beispielcodefragment in das Projektformular ein. Das Codefragment unten validiert eine XML-Datei. Die in diesem Codefragment verwendete XML-Datei befindet sich im AltovaXML-Applikationsordner im Ordner `AltovaXMLExamples`.
4. Kompilieren Sie den Code und testen Sie ihn.

```
// Locate examples installed with AltovaXML
// REMARK: You might need to adapt this if you have a different major version
of the product (2011 in this example).
   String strExamplesFolder = Environment.GetEnvironmentVariable
("ProgramFiles") + "\\Altova\\AltovaXML2011\\AltovaXMLExamples\\";

// Create a new AltovaXML instance and access its engines
   Altova.AltovaXML.Application AltovaXML = new Altova.AltovaXML.Application
();

// Validate input file simple.xml - it must be well-formed but not necessarily
valid.
// The AltovaXML application will provide us with a validator object.
   Altova.AltovaXML.XMLValidator AltovaXMLValidator = AltovaXML.XMLValidator;
```

```
    AltovaXMLValidator.InputXMLFileName = strExamplesFolder + "simple.xml";
    Boolean bIsWellFormed = AltovaXMLValidator.IsWellFormed();
    Boolean bIsValid = AltovaXMLValidator.IsValid();

// Show result
    MessageBox.Show("File " + strExamplesFolder + "simple.xml" + " is " +
       (bIsWellFormed ? "well-formed" : "not Well-formed") +
       " and " + (bIsValid ? "valid" : "invalid") + ".");
```


© 2012 Altova GmbH

2.4.3 Altova.AltovaXML.XSLT1

Beschreibung
Das `Altova.AltovaXML.XSLT1` Objekt stellt Methoden und Eigenschaften zur Ausführung einer XSLT 1.0-Transformation mit Hilfe des Altova XSLT 1.0-Prozessors bereit. Die Ergebnisse können in einer Datei gespeichert werden oder als String zurückgegeben werden. Mit Hilfe dieses Objekts können außerdem XSLT-Parameter an das XSLT Stylesheet übergeben werden. Die URLs von XML und XSLT-Dateien können über die Eigenschaften des Objekts bereitgestellt werden. Alternativ dazu können das XML- und das XSLT-Dokument auch innerhalb des Codes als Textstrings erstellt werden.

Anmerkung: Wenn Eingabestrings als URLs interpretiert werden sollen, sollten absolute Pfade verwendet werden. Bei Verwendung eines relativen Pfads sollte im aufrufenden Modul ein Mechanismus zur Auflösung des relativen Pfads definiert sein.

Methoden
Die folgenden Methoden stehen zur Verfügung:

`Execute(``OutputFileName as String`**`)`**
`void` **`execute`**`(String outputFilename)`
`Execute` führt die XSLT 1.0-Transformation aus und speichert das Ergebnis als Ausgabedatei, deren Name und Pfad als Input-String für die `Execute` Methode bereitgestellt wird. z.B.:
`Execute("C:\OutputDoc.xml").`

`ExecuteAndGetResultAsString()` `as String`
`ExecuteAndGetResultAsString` führt eine XSLT 1.0 Transformation aus und gibt das Ergebnis als UTF 16-Textstring zurück. *Beispiele siehe unten.*

`AddExternalParameter(``ParamName as String, ParamValue as String`**`)`**
Nimmt einen Parameternamen und den Wert dieses Parameters als Input-Argumente. Die einzelnen externen Parameter und deren Werte müssen in einem separaten Aufruf der Methode definiert werden. Wenn derselbe Parametername in mehrere Aufrufen definiert ist, wird der im letzten Aufruf definierte Wert verwendet. Da Parameterwerte XPath-Ausdrücke sind, müssen Parameterwerte, die Strings sind, in einfache Anführungszeichen eingeschlossen werden. In diesem Beispiel werden zwei Parameterwerte verwendet:

```
AddExternalParameter("Param1","'http://www.altova.com/'");
AddExternalParameter("Param2","concat('http://www.altova.com/',
MyFile/@url)");
```

`ClearExternalParameterList()`
Es sollte kein Argument bereitgestellt werden. Die Methode `ClearExternalParameterList` löscht die Liste der externen Parameter, die mit `AddExternalParameter` Methoden erstellt wurden.

Hinweis: Die Transformation muss immer erfolgen, nachdem das XML-Dokument und das XSLT-Dokument zugewiesen wurden.

Eigenschaften
Es sind die folgenden Eigenschaften definiert:

`InputXMLFileName`
Ein String-Input, der als URL gelesen wird, über die die zu transformierende XML-Datei gefunden wird.

`XSLFileName`

Ein String-Input, der als URL zum Auffinden der XSLT-Datei gelesen wird, die für die Transformation verwendet werden soll.

`InputXMLFromText`
Ein String-Input, der ein XML-Dokument erstellt.

`XSLFromText`
Ein String-Input, der ein XSLT-Dokument erstellt.

`XSLStackSize`
Die Stack-Größe ist die maximale Tiefe, bis zu der Anweisungen ausgeführt werden sollen. Die Stack-Größe kann über die Eigenschaft `XSLStackSize` geändert werden. Die zulässige Mindest-Stack-Größe ist 100. Die Standard-Stack-Größe ist 1000. Bei Überschreitung der Stack-Größe während einer Transformation wird ein Fehler ausgegeben.

`LastErrorMessage`
Gibt die letzte Fehlermeldung zurück.

`JavaExtensionsEnabled`
Aktiviert die Java-Erweiterungen. Sie können festlegen, ob Java-Erweiterungen aktiviert werden sollen oder nicht, indem Sie den Wert `true` oder `false` (keine Berücksichtigung der Groß- und Kleinschreibung) als Argument Variant_Bool liefern.

`DotNetExtensionsEnabled`
Aktiviert die .NET-Erweiterungen. Sie können festlegen, ob .NET-Erweiterungen aktiviert werden sollen oder nicht, indem Sie den Wert `true` oder `false` (keine Berücksichtigung der Groß- und Kleinschreibung) als Argument Variant_Bool liefern.

Beispiele
In den folgenden C#-Codefragmenten wird gezeigt wie man:

- ein XML-Dokument validiert und eine XSLT 1.0-Transformation ausführt (Transformation aus der XML-Datei in String)
- eine Transformation mittels XSLT 1.0 (XML-Datei in XML-Datei) durchführt
- eine Transformation mittels XSLT 2.0 (String in XML-Datei) durchführt
- eine Transformation mittels XSLT 1.0 (String in String) durchführt
-

Ein ausführlicheres Beispiel finden Sie am Ende des Abschnitts .NET-Schnittstelle.

Um diese Codefragmente in einem C#-Projekt zu erstellen, gehen Sie folgendermaßen vor:

1. Fügen Sie in Microsoft Visual Studio mit dem Befehl **Datei | Neu | Projekt** ein neues Projekt hinzu.
2. Fügen Sie durch Klicken auf **Projekt | Referenz hinzufügen** eine Referenz zur AltovaXML DLL hinzu. Daraufhin wird das Dialogfeld "Referenz hinzufügen" angezeigt, in dem eine Liste der installierten .NET-Komponenten angezeigt wird. Wählen Sie darin die AltovaXML-Komponente aus um sie hinzuzufügen. (Anmerkung: Wenn sich die AltovaXML-Komponente nicht in der Liste auf dem Register .NET befindet, kann sie vom Register COM ausgewählt werden.)
3. Geben Sie das unten gezeigte Beispielcodefragment in das Projektformular ein. Das Codefragment unten validiert eine XML-Datei und führt eine XSLT 1.0-Transformation an der XML-Datei durch. Die in diesem Codefragment verwendete XML-Datei befindet sich im AltovaXML-Applikationsordner im Ordner `AltovaXMLExamples`.
4. Kompilieren Sie den Code und testen Sie ihn.

Validierung und XSLT 1.0-Transformation (XML in String)

```
// Specify folder (AltovaXMLExamples folder)
```

 © 2012 Altova GmbH

```
// Check if filepath is correct for you
   String strExamplesFolder = Environment.GetEnvironmentVariable
("ProgramFiles") + "\\Altova\\AltovaXML2011\\AltovaXMLExamples\\";

// Create a new AltovaXML instance and access its engines
   Altova.AltovaXML.Application AltovaXML = new Altova.AltovaXML.Application
();

// Use Validator of AltovaXML to validate input file simple.xml
// File must be well-formed but not necessarily valid
   Altova.AltovaXML.XMLValidator AltovaXMLValidator = AltovaXML.XMLValidator;
   AltovaXMLValidator.InputXMLFileName = strExamplesFolder + "simple.xml";
   Boolean bIsWellFormed = AltovaXMLValidator.IsWellFormed();
   Boolean bIsValid = AltovaXMLValidator.IsValid();

// Show result
   MessageBox.Show("File " + strExamplesFolder + "simple.xml" + " is " +
      (bIsWellFormed ? "well-formed" : "not Well-formed") +
      " and " + (bIsValid ? "valid" : "invalid") + ".");

   if (bIsWellFormed)
   {
      // Use XSLT1 Engine of AltovaXML to transform simple.xml using
CopyInputXSLT1.xsl
         Altova.AltovaXML.IXSLT1 AltovaXMLXSLT1 = AltovaXML.XSLT1;
         AltovaXMLXSLT1.InputXMLFileName = strExamplesFolder + "simple.xml";
         AltovaXMLXSLT1.XSLFileName = strExamplesFolder + "CopyInputXSLT1.
xsl";
         String strResult = AltovaXMLXSLT1.ExecuteAndGetResultAsString();

      // Show result
         MessageBox.Show("XSLT 1.0 engine answered: " + strResult);
   }
```

XSLT 1.0-Transformation (XML in XML)

```
// Specify folder (AltovaXMLExamples folder)
// Check if filepath is correct for you
   String strExamplesFolder = Environment.GetEnvironmentVariable
("ProgramFiles") + "\\Altova\\AltovaXML2011\\AltovaXMLExamples\\";

// Create a new AltovaXML instance and access its engines
   Altova.AltovaXML.Application AltovaXML = new Altova.AltovaXML.Application
();

// Use XSLT1 Engine of AltovaXML to transform simple.xml using CopyInputXSLT1.
xsl
   Altova.AltovaXML.IXSLT1 AltovaXMLXSLT1 = AltovaXML.XSLT1;
   AltovaXMLXSLT1.InputXMLFileName = strExamplesFolder + "simple.xml";
   AltovaXMLXSLT1.XSLFileName = strExamplesFolder + "CopyInputXSLT1.xsl";
   AltovaXMLXSLT1.Execute(strExamplesFolder + "simpleOutputFromXML.xml");
```

XSLT 1.0-Transformation (String in XML)

```
// Specify folder (AltovaXMLExamples folder)
// Check if filepath is correct for you
   String strExamplesFolder = Environment.GetEnvironmentVariable
("ProgramFiles") + "\\Altova\\AltovaXML2011\\AltovaXMLExamples\\";

// Create a new AltovaXML instance and access its engines
   Altova.AltovaXML.Application AltovaXML = new Altova.AltovaXML.Application
();
```

```
// Use XSLT1 Engine of AltovaXML to transform input string using
CopyInputXSLT1.xsl
   Altova.AltovaXML.IXSLT1 AltovaXMLXSLT1 = AltovaXML.XSLT1;
   AltovaXMLXSLT1.InputXMLFromText = "<?xml version='1.0'?><doc>Hello
World</doc>";
   AltovaXMLXSLT1.XSLFileName = strExamplesFolder + "CopyInputXSLT1.xsl";
   AltovaXMLXSLT1.Execute(strExamplesFolder + "simpleOutputFromString.xml");
```

XSLT 1.0-Transformation (String in String)

```
// Specify folder (AltovaXMLExamples folder)
// Check if filepath is correct for you
   String strExamplesFolder = Environment.GetEnvironmentVariable
("ProgramFiles") + "\\Altova\\AltovaXML2011\\AltovaXMLExamples\\";

// Create a new AltovaXML instance and access its engines
   Altova.AltovaXML.Application AltovaXML = new Altova.AltovaXML.Application
();

// Use XSLT1 Engine of AltovaXML to transform input string using
CopyInputXSLT1.xsl
   Altova.AltovaXML.IXSLT1 AltovaXMLXSLT1 = AltovaXML.XSLT1;
   AltovaXMLXSLT1.InputXMLFromText = "<?xml version='1.0'?><doc>Hello
World</doc>";
   AltovaXMLXSLT1.XSLFileName = strExamplesFolder + "CopyInputXSLT1.xsl";
   String strResult = AltovaXMLXSLT1.ExecuteAndGetResultAsString();

// Show result
   MessageBox.Show("XSLT 1.0 engine answered: " + strResult);
```

Verwendung von .NET-Erweiterungen

```
// Specify folder (AltovaXMLExamples folder)
// Check if filepath is correct for you
   String strExamplesFolder = Environment.GetEnvironmentVariable
("ProgramFiles") + "\\Altova\\AltovaXML2011\\AltovaXMLExamples\\";

// Create a new AltovaXML instance and access its engines
   Altova.AltovaXML.Application AltovaXML = new Altova.AltovaXML.Application
();

// Use XSLT1 Engine from AltovaXML application
   Altova.AltovaXML.IXSLT1 AltovaXMLXSLT1 = AltovaXML.XSLT1;

// Enable .NET extensions
   AltovaXMLXSLT1.DotNetExtensionsEnabled = 1;

// Use XSLT containing .NET math extension for transformation
   AltovaXMLXSLT1.InputXMLFileName = strExamplesFolder + "simple.xml";
   AltovaXMLXSLT1.XSLFromText = "<xsl:stylesheet
xmlns:xsl='http://www.w3.org/1999/XSL/Transform'
      xmlns:math='clitype:System.Math' version='1.0'><xsl:output
omit-xml-declaration='yes'/>
      <xsl:template match='/'><a><sqrtanswer><xsl:value-of select='
math:Sqrt(9)'/></sqrtanswer></a>
      </xsl:template></xsl:stylesheet>";
   AltovaXMLXSLT1.Execute(strExamplesFolder + "Output.xml");

// Release ALL references to all components that were received.
   System.Runtime.InteropServices.Marshal.ReleaseComObject(AltovaXMLXSLT1);
```

 © 2012 Altova GmbH

```
   AltovaXMLXSLT1 = null;
   System.Runtime.InteropServices.Marshal.ReleaseComObject(AltovaXML);
    AltovaXML = null;
```

Verwendung der Eigenschaft `LastErrorMessage`

```
// Specify folder (AltovaXMLExamples folder)
// Check if filepath is correct for you
   String strExamplesFolder = Environment.GetEnvironmentVariable
("ProgramFiles") + "\\Altova\\AltovaXML2011\\AltovaXMLExamples\\";

// Create a new AltovaXML instance and access its engines
   Altova.AltovaXML.Application AltovaXML = new Altova.AltovaXML.Application
();

// Use XSLT1 Engine from AltovaXML application
   Altova.AltovaXML.IXSLT1 AltovaXMLXSLT1 = AltovaXML.XSLT1;

// Enable/disable .NET extensions (true/false, 1/0)
   AltovaXMLXSLT1.DotNetExtensionsEnabled = 0;

// Use XSLT containing .NET math extension for transformation
   AltovaXMLXSLT1.InputXMLFileName = strExamplesFolder + "simple.xml";
   AltovaXMLXSLT1.XSLFromText = "<xsl:stylesheet
xmlns:xsl='http://www.w3.org/1999/XSL/Transform'
      xmlns:math='clitype:System.Math' version='1.0'><xsl:output
omit-xml-declaration='yes'/>
      <xsl:template match='/'><a><sqrtanswer><xsl:value-of select='
math:Sqrt(9)'/></sqrtanswer></a>
      </xsl:template></xsl:stylesheet>";
   try
   {
      AltovaXMLXSLT1.Execute(strExamplesFolder + "Output.xml");
   }
   catch (Exception)
   {
      String strError = AltovaXMLXSLT1.LastErrorMessage;
      // Show result
      MessageBox.Show("XSLT 1.0 engine errors: " + strError);
   }

// Release ALL references to all components that were received.
   System.Runtime.InteropServices.Marshal.ReleaseComObject(AltovaXMLXSLT1);
   AltovaXMLXSLT1 = null;
   System.Runtime.InteropServices.Marshal.ReleaseComObject(AltovaXML);
   AltovaXML = null;
```

2.4.4 Altova.AltovaXML.XSLT2

Beschreibung
Das `Altova.AltovaXML.XSLT2` Objekt stellt Methoden und Eigenschaften zum Ausführen einer XSLT 2.0-Transformation mittels des Altova XSLT 2.0-Prozessors bereit. Das Ergebnis kann in einer Datei oder als String zurückgegeben werden. Über das Objekt können außerdem XSLT-Parameter an das XSLT-Stylesheet übergeben werden. Die URLs von XML- und XSLT-Dateien können als Strings über Eigenschaften des Objekts bereitgestellt werden. Alternativ dazu können XML- und XSLT-Dokumente innerhalb Codes als Textstrings konstruiert werden.

Hinweis:
- Wo String-Inputs als URLs interpretiert werden sollen, sollten absolute Pfade verwendet werden. Bei Verwendung eines relativen Pfads sollte im aufrufenden Modul ein Mechanismus definiert werden, der den relativen Pfad auflöst.
- Der XSLT 2.0-Prozessor kann zur Verarbeitung eines XSLT 1.0 Stylesheet im Rückwärtskompatibilitätsmodus verwendet werden. Die Ausgabe kann sich allerdings von der eines XSLT 1.0-Prozessors, der dasselbe XSLT 1.0 Stylesheet verarbeitet, unterscheiden.

Methoden
Es stehen die folgenden Methoden zur Verfügung:

`Execute(``OutputFileName as String`**`)`**
`void` **`execute`**`(String outputFilename)`
`Execute` führt die XSLT 2.0-Transformation aus und speichert das Ergebnis als Ausgabedatei, deren Name und Pfad als Input-String für die `Execute` Methode bereitgestellt wird. z.B.: `Execute("C:\OutputDoc.xml").`

`ExecuteAndGetResultAsString()` `as String`
`ExecuteAndGetResultAsString` führt eine XSLT 2.0 Transformation aus und gibt das Ergebnis als UTF 16-Textstring zurück. *Beispiele siehe unten.*

`AddExternalParameter(``ParamName as String, ParamValue as String`**`)`**
Nimmt einen Parameternamen und den Wert dieses Parameters als Input-Argumente. Die einzelnen externen Parameter und deren Werte müssen in einem separaten Aufruf der Methode definiert werden. Wenn derselbe Parametername in mehrere Aufrufen definiert ist, wird der im letzten Aufruf definierte Wert verwendet. Da Parameterwerte XPath-Ausdrücke sind, müssen Parameterwerte, die Strings sind, in einfache Anführungszeichen eingeschlossen werden. In diesem Beispiel werden zwei Parameterwerte verwendet:

```
AddExternalParameter("Param1","'http://www.altova.com/'");
AddExternalParameter("Param2","concat('http://www.altova.com/',
MyFile/@url)");
```

`ClearExternalParameterList()`
Es sollte kein Argument bereitgestellt werden. Die Methode `ClearExternalParameterList` löscht die Liste der externen Parameter, die mit `AddExternalParameter` Methoden erstellt wurden.

`InitialTemplateName`
Definiert die anfangs verwendete named Template. Das Argument ist der Name der Vorlage, an der die Verarbeitung beginnen soll. z.B. `InitialNamedTemplat="MyNamedTemplate"`.

`InitialTemplateMode`
Definiert den anfangs verwendeten Modus für die Verarbeitung. Das Argument ist der Name

 © 2012 Altova GmbH

des Anfangsmodus. Vorlagen mit diesem Moduswert werden verarbeitet. z.B. `InitialTemplateMode="MyMode"`.

Hinweis: Die Transformation muss immer erfolgen, nachdem das XML-Dokument und das XSLT-Dokument zugewiesen wurden.

Eigenschaften
Es sind die folgenden Eigenschaften definiert:

`InputXMLFileName`
Ein String-Input, der als URL gelesen wird, über die die zu transformierende XML-Datei gefunden wird.

`XSLFileName`
Ein String-Input, der als URL zum Auffinden der XSLT-Datei gelesen wird, die für die Transformation verwendet werden soll.

`InputXMLFromText`
Ein String-Input, der ein XML-Dokument erstellt.

`XSLFromText`
Ein String-Input, der ein XSLT-Dokument erstellt.

`XSLStackSize`
Die Stack-Größe ist die maximale Tiefe, bis zu der Anweisungen ausgeführt werden sollen. Die Stack-Größe kann über die Eigenschaft `XSLStackSize` geändert werden. Die zulässige Mindest-Stack-Größe ist 100. Die Standard-Stack-Größe ist 1000. Bei Überschreitung der Stack-Größe während einer Transformation wird ein Fehler ausgegeben.

`LastErrorMessage`
Gibt die letzte Fehlermeldung zurück.

`JavaExtensionsEnabled`
Aktiviert die Java-Erweiterungen. Sie können festlegen, ob Java-Erweiterungen aktiviert werden sollen oder nicht, indem Sie den Wert `true` oder `false` (keine Berücksichtigung der Groß- und Kleinschreibung) als Argument Variant_Bool liefern.

`DotNetExtensionsEnabled`
Aktiviert die .NET-Erweiterungen. Sie können festlegen, ob .NET-Erweiterungen aktiviert werden sollen oder nicht, indem Sie den Wert `true` oder `false` (keine Berücksichtigung der Groß- und Kleinschreibung) als Argument Variant_Bool liefern.

Beispiele
In den folgenden C#-Codefragmenten wird gezeigt wie man:

- ein XML-Dokument validiert und eine XSLT 1.0-Transformation ausführt (Transformation aus der XML-Datei in String)
- eine Transformation mittels XSLT 1.0 (XML-Datei in XML-Datei) durchführt
- eine Transformation mittels XSLT 2.0 (String in XML-Datei) durchführt
- eine Transformation mittels XSLT 1.0 (String in String) durchführt
-

Ein ausführlicheres Beispiel finden Sie am Ende des Abschnitts .NET-Schnittstelle.

Um diese Codefragmente in einem C#-Projekt zu erstellen, gehen Sie folgendermaßen vor:

1. Fügen Sie in Microsoft Visual Studio mit dem Befehl **Datei | Neu | Projekt** ein neues Projekt hinzu.
2. Fügen Sie durch Klicken auf **Projekt | Referenz hinzufügen** eine Referenz zur

AltovaXML DLL hinzu. Daraufhin wird das Dialogfeld "Referenz hinzufügen" angezeigt, in dem eine Liste der installierten .NET-Komponenten angezeigt wird. Wählen Sie darin die AltovaXML-Komponente aus um sie hinzuzufügen. (Anmerkung: Wenn sich die AltovaXML-Komponente nicht in der Liste auf dem Register .NET befindet, kann sie vom Register COM ausgewählt werden.)

3. Geben Sie das unten gezeigte Beispielcodefragment in das Projektformular ein. Das Codefragment unten validiert eine XML-Datei und führt eine XSLT 2.0-Transformation an der XML-Datei durch. Die in diesem Codefragment verwendete XML-Datei befindet sich im AltovaXML-Applikationsordner im Ordner `AltovaXMLExamples`.
4. Kompilieren Sie den Code und testen Sie ihn.

Validierung und XSLT 2.0-Transformation (XML in string)

```
// Specify folder (AltovaXMLExamples folder)
// Check if filepath is correct for you
   String strExamplesFolder = Environment.GetEnvironmentVariable
("ProgramFiles") + "\\Altova\\AltovaXML2011\\AltovaXMLExamples\\";

// Create a new AltovaXML instance and access its engines
   Altova.AltovaXML.Application AltovaXML = new Altova.AltovaXML.Application
();

// Use Validator of AltovaXML to validate input file simple.xml
// File must be well-formed but not necessarily valid
   Altova.AltovaXML.XMLValidator AltovaXMLValidator = AltovaXML.XMLValidator;
   AltovaXMLValidator.InputXMLFileName = strExamplesFolder + "simple.xml";
   Boolean bIsWellFormed = AltovaXMLValidator.IsWellFormed();
   Boolean bIsValid = AltovaXMLValidator.IsValid();

// Show result
   MessageBox.Show("File " + strExamplesFolder + "simple.xml" + " is " +
      (bIsWellFormed ? "well-formed" : "not Well-formed") +
      " and " + (bIsValid ? "valid" : "invalid") + ".");

   if (bIsWellFormed)
   {
      // Use XSLT2 Engine of AltovaXML to transform simple.xml using
CopyInputXSLT2.xsl
         Altova.AltovaXML.IXSLT2 AltovaXMLXSLT2 = AltovaXML.XSLT2;
         AltovaXMLXSLT2.InputXMLFileName = strExamplesFolder + "simple.xml";
         AltovaXMLXSLT2.XSLFileName = strExamplesFolder + "CopyInputXSLT2.
xsl";
         String strResult = AltovaXMLXSLT2.ExecuteAndGetResultAsString();

      // Show result
         MessageBox.Show("XSLT 2.0 engine answered: " + strResult);
   }
```

XSLT 2.0-Transformation (XML in XML)

```
// Specify folder (AltovaXMLExamples folder)
// Check if filepath is correct for you
   String strExamplesFolder = Environment.GetEnvironmentVariable
("ProgramFiles") + "\\Altova\\AltovaXML2011\\AltovaXMLExamples\\";

// Create a new AltovaXML instance and access its engines
   Altova.AltovaXML.Application AltovaXML = new Altova.AltovaXML.Application
();

// Use XSLT2 Engine of AltovaXML to transform simple.xml using CopyInputXSLT2.
xsl
```

 © 2012 Altova GmbH

```
   Altova.AltovaXML.IXSLT2 AltovaXMLXSLT2 = AltovaXML.XSLT2;
   AltovaXMLXSLT2.InputXMLFileName = strExamplesFolder + "simple.xml";
   AltovaXMLXSLT2.XSLFileName = strExamplesFolder + "CopyInputXSLT2.xsl";
   AltovaXMLXSLT2.Execute(strExamplesFolder + "simpleOutputFromXML.xml");
```

XSLT 2.0-Transformation (String in XML)

```
// Specify folder (AltovaXMLExamples folder)
// Check if filepath is correct for you
   String strExamplesFolder = Environment.GetEnvironmentVariable
("ProgramFiles") + "\\Altova\\AltovaXML2011\\AltovaXMLExamples\\";

// Create a new AltovaXML instance and access its engines
   Altova.AltovaXML.Application AltovaXML = new Altova.AltovaXML.Application
();

// Use XSLT2 Engine of AltovaXML to transform input string using
CopyInputXSLT2.xsl
   Altova.AltovaXML.IXSLT2 AltovaXMLXSLT2 = AltovaXML.XSLT2;
   AltovaXMLXSLT2.InputXMLFromText = "<?xml version='1.0'?><doc>Hello
World</doc>";
   AltovaXMLXSLT2.XSLFileName = strExamplesFolder + "CopyInputXSLT2.xsl";
   AltovaXMLXSLT2.Execute(strExamplesFolder + "simpleOutputFromString.xml");
```

XSLT 2.0-Transformation (String in String)

```
// Specify folder (AltovaXMLExamples folder)
// Check if filepath is correct for you
   String strExamplesFolder = Environment.GetEnvironmentVariable
("ProgramFiles") + "\\Altova\\AltovaXML2011\\AltovaXMLExamples\\";

// Create a new AltovaXML instance and access its engines
   Altova.AltovaXML.Application AltovaXML = new Altova.AltovaXML.Application
();

// Use XSLT2 Engine of AltovaXML to transform input string using
CopyInputXSLT2.xsl
   Altova.AltovaXML.IXSLT2 AltovaXMLXSLT2 = AltovaXML.XSLT2;
   AltovaXMLXSLT2.InputXMLFromText = "<?xml version='1.0'?><doc>Hello
World</doc>";
   AltovaXMLXSLT2.XSLFileName = strExamplesFolder + "CopyInputXSLT2.xsl";
   String strResult = AltovaXMLXSLT2.ExecuteAndGetResultAsString();

// Show result
   MessageBox.Show("XSLT 2.0 engine answered: " + strResult);
```

Verwendung von .NET-Erweiterungen

```
// Specify folder (AltovaXMLExamples folder)
// Check if filepath is correct for you
   String strExamplesFolder = Environment.GetEnvironmentVariable
("ProgramFiles") + "\\Altova\\AltovaXML2011\\AltovaXMLExamples\\";

// Create a new AltovaXML instance and access its engines
   Altova.AltovaXML.Application AltovaXML = new Altova.AltovaXML.Application
();
```

```
// Use XSLT2 Engine from AltovaXML application
   Altova.AltovaXML.IXSLT2 AltovaXMLXSLT2 = AltovaXML.XSLT2;

// Enable .NET extensions
   AltovaXMLXSLT2.DotNetExtensionsEnabled = 1;

// Use XSLT containing .NET math extension for transformation
   AltovaXMLXSLT2.InputXMLFileName = strExamplesFolder + "simple.xml";
   AltovaXMLXSLT2.XSLFromText = "<xsl:stylesheet
xmlns:xsl='http://www.w3.org/1999/XSL/Transform'
      xmlns:math='clitype:System.Math' version='2.0'><xsl:output
omit-xml-declaration='yes'/>
      <xsl:template match='/'><a><sqrtanswer><xsl:value-of select='
math:Sqrt(9)'/></sqrtanswer></a>
      </xsl:template></xsl:stylesheet>";
   AltovaXMLXSLT2.Execute(strExamplesFolder + "Output.xml");

// Release ALL references to all components that were received.
   System.Runtime.InteropServices.Marshal.ReleaseComObject(AltovaXMLXSLT2);
   AltovaXMLXSLT2 = null;
   System.Runtime.InteropServices.Marshal.ReleaseComObject(AltovaXML);
   AltovaXML = null;
```

Verwendung der Eigenschaft `LastErrorMessage`

```
// Specify folder (AltovaXMLExamples folder)
// Check if filepath is correct for you
   String strExamplesFolder = Environment.GetEnvironmentVariable
("ProgramFiles") + "\\Altova\\AltovaXML2011\\AltovaXMLExamples\\";

// Create a new AltovaXML instance and access its engines
   Altova.AltovaXML.Application AltovaXML = new Altova.AltovaXML.Application
();

// Use XSLT2 Engine from AltovaXML application
   Altova.AltovaXML.IXSLT2 AltovaXMLXSLT2 = AltovaXML.XSLT2;

// Enable/disable .NET extensions (true/false, 1/0)
   AltovaXMLXSLT2.DotNetExtensionsEnabled = 0;

// Use XSLT containing .NET math extension for transformation
   AltovaXMLXSLT2.InputXMLFileName = strExamplesFolder + "simple.xml";
   AltovaXMLXSLT2.XSLFromText = "<xsl:stylesheet
xmlns:xsl='http://www.w3.org/1999/XSL/Transform'
      xmlns:math='clitype:System.Math' version='2.0'><xsl:output
omit-xml-declaration='yes'/>
      <xsl:template match='/'><a><sqrtanswer><xsl:value-of select='
math:Sqrt(9)'/></sqrtanswer></a>
      </xsl:template></xsl:stylesheet>";
   try
   {
      AltovaXMLXSLT2.Execute(strExamplesFolder + "Output.xml");
   }
   catch (Exception)
   {
      String strError = AltovaXMLXSLT2.LastErrorMessage;
      // Show result
      MessageBox.Show("XSLT 2.0 engine errors: " + strError);
   }

// Release ALL references to all components that were received.
   System.Runtime.InteropServices.Marshal.ReleaseComObject(AltovaXMLXSLT2);
```


© 2012 Altova GmbH

```
AltovaXMLXSLT2 = null;
System.Runtime.InteropServices.Marshal.ReleaseComObject(AltovaXML);
AltovaXML = null;
```

2.4.5 Altova.AltovaXML.XQuery

Beschreibung
Das `Altova.AltovaXML.XQuery` Objekt stellt Methoden und Eigenschaften zum Ausführen einer XQuery 1.0-Transformation mittels des Altova XQuery 1.0-Prozessors bereit. Das Ergebnis kann in einer Datei gespeichert oder als String zurückgegeben werden. Über das Objekt können außerdem XQuery-Variablen an das XQuery-Dokument übergeben werden. Die URLs von XQuery- und XML-Dateien können als Strings über Eigenschaften des Objekts bereitgestellt werden. Alternativ dazu können das XML- und das XQuery-Dokument innerhalb des Codes als Textstrings konstruiert werden.

Hinweis: Wo String-Inputs als URLs interpretiert werden sollen, sollten absolute Pfade verwendet werden. Bei Verwendung eines relativen Pfads sollte im aufrufenden Modul ein Mechanismus definiert werden, der den relativen Pfad auflöst.

Methoden
Es stehen die folgenden Methoden zur Verfügung:

`Execute`
`void` **`execute`**`(String outputFilename)`
`Execute` führt die XQuery 1.0-Transformation aus und speichert das Ergebnis als Ausgabedatei, deren Name und Pfad als Input-String für die `Execute` Methode bereitgestellt werden. z.B.: `Execute("C:\OutputDoc.xml").`

`ExecuteAndGetResultAsString()` `as String`
`ExecuteAndGetResultAsString` führt eine XQuery 1.0-Transformation aus und gibt das Ergebnis als UTF 16- Textstring zurück. *Beispiele siehe unten.*
`AddExternalVariable(``VarName as String, VarValue as String`**`)`**
Nimmt einen Variablennamen und den Wert dieser Variable als Input-Argumente. Die einzelnen externen Variablen und deren Werte müssen in einem separaten Aufruf der Methode definiert werden. Variablen müssen im XQuery-Dokument deklariert werden, optional mit einer Typdeklaration. Unabhängig von der Typdeklaration für die externe Variable im XQuery-Dokument sind für den Variablenwert, der für die Methode `AddExternalVariable` bereitgestellt wird, keine speziellen Trennzeichen wie z.B. Anführungszeichen erforderlich. Die lexikalische Form muss jedoch der des erwarteten Typs entsprechen (so muss z.B. eine Variable vom Typ `xs:date` einen Wert in der lexikalischen Form `2004-01-31` haben; ein Wert in der lexikalischen Form `2004/Jan/01` verursacht dagegen einen Fehler). Beachten Sie, dass dies auch bedeutet, dass Sie eine XQuery 1.0-Funktion (z.B. `current-date()`) nicht als den Wert einer externen Variable verwenden können (da die lexikalische Form der Funktion in der eigenen Schreibweise entweder nicht dem erforderlichen Datentyp entspricht - wenn der Datentyp in der Deklaration der externen Variable definiert ist - oder als String gelesen wird, wenn der Datentyp nicht definiert ist. Wenn eine externe Variable bereitgestellt wird, die den Namen einer existierenden (nicht gelöschten) Variable hat, so wird ein Fehler ausgegeben. Wenn derselbe Variablenname in mehrere Aufrufen definiert ist, wird der im letzten Aufruf definierte Wert verwendet.

`ClearExternalVariableList()`
Es sollte kein Argument bereitgestellt werden. Die Methode `ClearExternalVariableList` löscht die Liste der externen Variablen, die mit `AddExternalVariable` Methoden erstellt wurden.

Hinweis: Die Definition des optionalen XML-Dokuments muss immer vor der XQuery-Ausführung erfolgen.

Eigenschaften

 © 2012 Altova GmbH

Es sind die folgenden Eigenschaften definiert:

XQueryFileName
Ein String-Input, der als URL zum Auffinden der auszuführenden XQuery-Datei gelesen wird. Wenn sowohl die `XQueryFileName` Eigenschaft als auch die `QueryFromText` Eigenschaft definiert ist, so wird diejenige Eigenschaft verwendet, die in der Codesequenz als letzte definiert ist.

InputXMLFileName
Ein String-Input, der als URL zum Auffinden der in die Query zu ladende XML-Datei gelesen wird. XQuery Navigationsausdrücke werden in Beziehung auf den Dokumentnode dieses XML-Dokuments ausgewertet. Wenn sowohl die `InputFileName` Eigenschaft als auch die die `InputXMLFromText` Eigenschaft definiert ist, so wird diejenige Eigenschaft verwendet, die in der Codesequenz als letzte definiert ist.

XQueryFromText
Ein String-Input, der ein XQuery-Dokument erstellt. Wenn sowohl die `XQueryFileName` Eigenschaft als auch die die `XQueryFromText` Eigenschaft definiert ist, so wird diejenige Eigenschaft verwendet, die in der Codesequenz als letzte definiert ist.

InputXMLFromText
Ein String-Input, der ein XML-Dokument erstellt. XQuery Navigationsausdrücke werden in Bezug auf den Dokument-Node dieses XML-Dokuments ausgewertet. Wenn sowohl die `InputFileName` Eigenschaft als auch die die `InputXMLFromText` Eigenschaft definiert ist, so wird diejenige Eigenschaft verwendet, die in der Codesequenz als letzte definiert ist.

LastErrorMessage
Gibt die letzte Fehlermeldung zurück.

JavaExtensionsEnabled
Aktiviert die Java-Erweiterungen. Sie können festlegen, ob Java-Erweiterungen aktiviert werden sollen oder nicht, indem Sie den Wert `true` oder `false` (keine Berücksichtigung der Groß- und Kleinschreibung) als Argument Variant_Bool liefern.

DotNetExtensionsEnabled
Aktiviert die .NET-Erweiterungen. Sie können festlegen, ob .NET-Erweiterungen aktiviert werden sollen oder nicht, indem Sie den Wert `true` oder `false` (keine Berücksichtigung der Groß- und Kleinschreibung) als Argument Variant_Bool liefern.

Hinweis: Wenn ein XML-Dokument definiert ist und nicht für eine neue XQuery-Ausführung erforderlich ist, sollte dies durch Zuweisung eines leeren Strings gelöscht werden.

Es sind die folgenden Serialisierungsoptionen definiert:
Execute(`OutputFileName as String`**)**
Die benötigte Ausgabemethode kann durch Bereitstellung des erforderlichen Werts als String-Argument definiert werden. Gültige Werte sind: `xml`, `xhtml`, `html` und `text`. Z.B.: `objAltovaXML.XQuery.OutputMethod = "xml"`. Wenn der Wert ungültig ist, wird er ignoriert. Die Standardausgabemethode ist `xml`.

OutputOmitXMLDeclaration
Sie können angeben, ob die XML-Deklaration in der Ausgabe inkludiert werden soll oder nicht. Geben Sie dazu `true` oder `false` (Groß- und Kleinschreibung ist irrelevant) als Boolesches Argument an. Z.B: `objAltovaXML.XQuery.OutputOmitXMLDeclaration = "FALSE"`. Wenn der Wert ungültig ist, wird ein Fehler ausgegeben. Die Standardoption ist `TRUE`.

OutputIndent

Sie können festlegen, ob die Ausgabe Einrückungen enthalten soll oder nicht. Geben Sie dazu `true` oder `false` (Groß- und Kleinschreibung ist irrelevant) als Boolesches Argument an. Z.B.: `objAltovaXML.XQuery.OutputIndent = "TRUE"`. Wenn der Wert ungültig ist, wird ein Fehler ausgegeben. Die Standardoption ist False.

`OutputEncoding`
Die erforderliche Ausgabecodierung kann durch Angabe des Codierungswerts als String-Argument definiert werden. Z.B.: `objAltovaXML.XQuery.OutputEncoding = "UTF-8"`. Wenn der Wert ungültig ist, wird er ignoriert. Die Standardausgabecodierung ist UTF-8.

Hinweis: Bei den Serialisierungsoptionen gibt es Unterschiede bei der Verwendung der Raw-Schnittstelle und der Dispatch-Schnittstelle. Wenn bei Verwendung der Raw-Schnittstelle kein Argument mit diesen Eigenschaften bereitgestellt wird, so wird der aktuelle Wert der Eigenschaft zurückgegeben. Die Eingabe müsste in etwa so aussehen: `put_OutputOption( VARIANT_BOOL bVal )` bzw. `VARIANT_BOOL bVal = get_OutputOption()`, um Werte zu setzen und abzurufen. Bei der Dispatch-Schnittstelle können Sie `b = myXQuery.OutputOption` verwenden, um Werte abzurufen und `myXQuery.OutputOption = b` um Werte zu definieren. Bei der Dispatch-Schnittstelle würde z.B. `Sheet1.Cells(10, 2) = objAltovaXML.XQuery.OutputEncoding` die aktuelle Ausgabecodierung abrufen.

Beispiel
Im folgenden C#-Codefragment wird gezeigt, wie man ein XML-Dokument validiert. Ein ausführlicheres Beispiel finden Sie am Ende des Abschnitts .NET-Schnittstelle.

Um diese Codefragmente in einem C#-Projekt zu erstellen, gehen Sie folgendermaßen vor:

1. Fügen Sie in Microsoft Visual Studio mit dem Befehl **Datei | Neu | Projekt** ein neues Projekt hinzu.
2. Fügen Sie durch Klicken auf **Projekt | Referenz hinzufügen** eine Referenz zur AltovaXML DLL hinzu. Daraufhin wird das Dialogfeld "Referenz hinzufügen" angezeigt, in dem eine Liste der installierten .NET-Komponenten angezeigt wird. Wählen Sie darin die AltovaXML-Komponente aus um sie hinzuzufügen. (Anmerkung: Wenn sich die AltovaXML-Komponente nicht in der Liste auf dem Register .NET befindet, kann sie vom Register COM ausgewählt werden.)
3. Geben Sie das unten gezeigte Beispielcodefragment in das Projektformular ein. Das Codefragment unten validiert eine XML-Datei. Die in diesem Codefragment verwendete XML-Datei befindet sich im AltovaXML-Applikationsordner im Ordner `AltovaXMLExamples`.
4. Kompilieren Sie den Code und testen Sie ihn.

```
// Locate examples installed with AltovaXML
// REMARK: You might need to adapt this if you have a different major version
of the product (2011 in this example).
   String strExamplesFolder = Environment.GetEnvironmentVariable
("ProgramFiles") + "\\Altova\\AltovaXML2011\\AltovaXMLExamples\\";

// Create a new AltovaXML instance and access its engines
   Altova.AltovaXML.Application AltovaXML = new Altova.AltovaXML.Application
();

// Validate input file simple.xml - it must be well-formed but not necessarily
valid.
// The AltovaXML application will provide us with a validator object.
   Altova.AltovaXML.XMLValidator AltovaXMLValidator = AltovaXML.XMLValidator;
   AltovaXMLValidator.InputXMLFileName = strExamplesFolder + "simple.xml";
   Boolean bIsWellFormed = AltovaXMLValidator.IsWellFormed();
```

 © 2012 Altova GmbH

```
   Boolean bIsValid = AltovaXMLValidator.IsValid();

// Show result
   MessageBox.Show("File " + strExamplesFolder + "simple.xml" + " is " +
      (bIsWellFormed ? "well-formed" : "not Well-formed") +
      " and " + (bIsValid ? "valid" : "invalid") + ".");

   if (bIsWellFormed)
   {
      // use XQuery Engine from AltovaXML application to transform simple.xml
with the help of CopyInput.xq
         Altova.AltovaXML.XQuery AltovaXMLXQuery = AltovaXML.XQuery;
         AltovaXMLXQuery.InputXMLFileName = strExamplesFolder + "simple.xml";
         AltovaXMLXQuery.XQueryFileName = strExamplesFolder + "CopyInput.xq";
         strResult = AltovaXMLXQuery.ExecuteAndGetResultAsString();

      // Show result
         MessageBox.Show("XQuery engine answered: " + strResult);
   }
```

2.4.6 Beispiel

Im folgenden C#-Codefragment wird gezeigt, wie man ein XML-Dokument validiert, XSLT 1.0- und XSLT 2.0-Transformationen durchführt und wie man ein XQuery-Dokument ausführt. Weitere Codefragmente finden Sie in den vorhergehenden Abschnitten zu den jeweiligen Prozessoren: XMLValidator; XSLT1; XSLT2; XQuery.

Um das Codefragment in einem C#-Projekt zu erstellen, gehen Sie folgendermaßen vor:

1. Fügen Sie in Microsoft Visual Studio mit dem Befehl **Datei | Neu | Projekt** ein neues Projekt hinzu.
2. Fügen Sie durch Auswahl des Befehls **Projekt | Verweis hinzufügen** hinzufügen eine Referenz zur AltovaXML Typbibliothek hinzu. Daraufhin wird das Dialogfeld "Verweis hinzufügen" angezeigt, in dem eine Liste der installierten COM-Komponenten angezeigt wird. Wählen Sie die AltovaXML Typbibliothekkomponente aus der Liste aus, um sie hinzuzufügen. (Beachten Sie, dass die AltovaXML-Komponente sich nicht in der Liste auf dem .NET-Register befindet. Sie kann vom Register "COM" ausgewählt werden.)
3. Geben Sie den gewünschten Code in das Projektformular ein. Das Codefragment unten validiert eine XML-Datei, führt unter Verwendung von XSLT 1.0 und XSLT 2.0 Stylesheets XSLT-Transformationen an der XML-Datei durch und führt ein XQuery-Dokument aus. Die in diesem Codefragment verwendeten Dateien befinden sich im AltovaXML Applikationsordner im Ordner `AltovaXMLExamples`.
4. Kompilieren Sie den Code und führen Sie ihn aus.

```
// Locate examples installed with AltovaXML
// REMARK: You might need to adapt this if you have a different major version
of the product (2011 in this example)
   String strExamplesFolder = Environment.GetEnvironmentVariable
("ProgramFiles") + "\\Altova\\AltovaXML2011\\AltovaXMLExamples\\";

// Create a new AltovaXML instance and access its engines
   Altova.AltovaXML.Application AltovaXML = new Altova.AltovaXML.Application
();

// Validate input file simple.xml - it must be well-formed but not necessarily
valid.
// The AltovaXML application will provide us with a validator object.
   Altova.AltovaXML.XMLValidator AltovaXMLValidator = AltovaXML.XMLValidator;
   AltovaXMLValidator.InputXMLFileName = strExamplesFolder + "simple.xml";
   Boolean bIsWellFormed = AltovaXMLValidator.IsWellFormed();
   Boolean bIsValid = AltovaXMLValidator.IsValid();

// Show result
   MessageBox.Show("File " + strExamplesFolder + "simple.xml" + " is " +
      (bIsWellFormed ? "well-formed" : "not Well-formed") +
      " and " + (bIsValid ? "valid" : "invalid") + ".");

// Release reference to XMLValidator component
   System.Runtime.InteropServices.Marshal
.ReleaseComObject(AltovaXMLValidator);
   AltovaXMLValidator = null;

   if (bIsWellFormed)
   {
      // Use XSLT1 Engine from the AltovaXML application to transform simple.
xml with the help of CopyInputXSLT1.xsl
         Altova.AltovaXML.IXSLT1 AltovaXMLXSLT1 = AltovaXML.XSLT1;
         AltovaXMLXSLT1.InputXMLFileName = strExamplesFolder + "simple.xml";
         AltovaXMLXSLT1.XSLFileName = strExamplesFolder + "CopyInputXSLT1.
xsl";
         String strResult = AltovaXMLXSLT1.ExecuteAndGetResultAsString();
         try
```

 © 2012 Altova GmbH

```
            {
               // Show result
               MessageBox.Show("XSLT 1.0 engine answered: " + strResult);
            }
            catch (Exception)
            {
               String strError = AltovaXMLXSLT1.LastErrorMessage;
               // Show errors
               MessageBox.Show("XSLT 1.0 engine errors: " + strError);
            }

         // Release reference to XMLXSLT1 component
            System.Runtime.InteropServices.Marshal
.ReleaseComObject(AltovaXMLXSLT1);
            AltovaXMLXSLT1 = null;

         // use XSLT2 Engine from AltovaXML application to transform simple.xml
with the help of CopyInputXSLT2.xsl
            Altova.AltovaXML.IXSLT2 AltovaXMLXSLT2 = AltovaXML.XSLT2;
            AltovaXMLXSLT2.InputXMLFileName = strExamplesFolder + "simple.xml";
            AltovaXMLXSLT2.XSLFileName = strExamplesFolder + "CopyInputXSLT2.
xsl";
            strResult = AltovaXMLXSLT2.ExecuteAndGetResultAsString();
            try
            {
               // Show result
               MessageBox.Show("XSLT 2.0 engine answered: " + strResult);
            }
            catch (Exception)
            {
               String strError = AltovaXMLXSLT2.LastErrorMessage;
               // Show errors
               MessageBox.Show("XSLT 2.0 engine errors: " + strError);
            }

         // Release reference to XMLXSLT2 component
            System.Runtime.InteropServices.Marshal
.ReleaseComObject(AltovaXMLXSLT2);
            AltovaXMLXSLT2 = null;

         // use XQuery Engine from AltovaXML application to transform simple.xml
with the help of CopyInput.xq
            Altova.AltovaXML.XQuery AltovaXMLXQuery = AltovaXML.XQuery;
            AltovaXMLXQuery.InputXMLFileName = strExamplesFolder + "simple.xml";
            AltovaXMLXQuery.XQueryFileName = strExamplesFolder + "CopyInput.xq";
            strResult = AltovaXMLXQuery.ExecuteAndGetResultAsString();
            try
            {
               // Show result
               MessageBox.Show("XQuery engine answered: " + strResult);
            }
            catch (Exception)
            {
               String strError = AltovaXMLXQuery.LastErrorMessage;
               // Show errors
               MessageBox.Show("XQuery engine errors: " + strError);
            }

         // Release reference to XMLXQuery component
            System.Runtime.InteropServices.Marshal
.ReleaseComObject(AltovaXMLXQuery);
            AltovaXMLXQuery = null;
      }

   // Release reference to AltovaXML component
      System.Runtime.InteropServices.Marshal.ReleaseComObject(AltovaXML);
```

```
AltovaXML = null;
```


© 2012 Altova GmbH

2.5 Explizite Freigabe von AltovaXML COM-Server von C# und VB.NETaus

Wenn Referenzen auf das COM Server-Objekt nicht freigegeben werden, wenn das Objekt außerhalb des Geltungsbereichs zu liegen kommt, können die AltovaXML COM-Referenzen explizit vom C#-Code aus mittels der ReleaseComObject-Methoden freigegeben werden, wie unten gezeigt.

Beispiel:

```
         private void button1_Click(object sender, EventArgs e)
        {
            Altova.AltovaXML.ApplicationClass AltovaXML = new Altova.AltovaXML
.ApplicationClass();
            Altova.AltovaXML.IXSLT2 XSLT2 = AltovaXML.XSLT2;

            XSLT2.InputXMLFileName =
"C:\\Projects\\files\\XMLSpyExeFolder\\Examples\\OrgChart.xml";
            XSLT2.XSLFileName =
"C:\\Projects\\files\\XMLSpyExeFolder\\Examples\\OrgChart.xsl";
            XSLT2.Execute(
"C:\\Projects\\files\\XMLSpyExeFolder\\Examples\\OrgChart_out.html");

            // Release the XSLT2 component and then the AltovaXML component
            System.Runtime.InteropServices.Marshal.ReleaseComObject(XSLT2);
            XSLT2 = null;
            System.Runtime.InteropServices.Marshal.ReleaseComObject(AltovaXML
);
            AltovaXML = null;
        }
```

- Am Ende der Methode wird der AltovaXML.exe **Server** beendet.
- Wenn Sie nicht **alle** ReleaseComObject Methoden aufrufen, werden die Exe-Server erst beim Beenden der C#-Applikation beendet.

2.6 OOXML- und ZIP-Dateien

Damit die erzeugten Dateien in Form einer ZIP-Datei oder Open Office XML (OOXML)-Dateien wie `.docx` ausgegeben werden, muss das ZIP-Protokoll im Dateipfad definiert werden. Beispiel:

```
filename.zip|zip/filename.xxx
filename.docx|zip/filename.xxx
```

In AltovaXML kann die Ausgabe in eine ZIP-Datei mit den folgenden Operationen definiert werden:

COM-Schnittstelle und .NET-Schnittstelle
Die Ausgabe wird mit Hilfe der `Execute` Methode generiert. Das Argument der Methode definiert den Namen und Pfad der Ausgabedatei. Für ZIP-Dateien muss das ZIP-Protokoll (siehe folgendes Beispiel) verwendet werden:

```
xslt2.Execute(c:\Mydocs\orgchart.zip|zip\main.xml)
xslt2.Execute(c:\Mydocs\orgchart.docx|zip\main.out)
xslt2.Execute(c:\Mydocs\orgchart.docx|zip\)
```

Befehlszeile
Stellen Sie bei Verwendung der Befehlszeile sicher, dass die Ausgabe-URI innerhalb von Anführungszeichen steht. Andernfalls würde das Pipe-Zeichen (|) vom Befehlszeilensystem interpretiert werden. Ein Beispiel:

```
AltovaXML -in input.xml -xslt2 transform.xslt -out "c:\results.zipart.zip|
zip\result.xml"
```

Das `xsl:result-document` Element
Im Falle des `xsl:result-document` Elements von XSLT 2.0 muss für die Ausgabe-URI das ZIP-Protokoll verwendet werden. Bei OOXML-Dokumenten muss das ZIP-Protokoll für die Ausgabe-URI jedes `xsl:result-document` Elements verwendet werden, das beim Erstellen von Dateien für das OOXML-Dokument verwendet wird. Wenn die `xsl:result-document` Elemente relative Ausgabe-URIs definieren, definieren Sie das ZIP-Protokoll für das Hauptergebnis, dessen URI anschließend als Basis-URI zum Auflösen der relativen Ausgabe-URIs dient.

© 2012 Altova GmbH

Kapitel 3

Prozessorinformationen

3 Prozessorinformationen

Dieser Abschnitt enthält Informationen über implementierungsspezifische Features des Altova XML Validators, des Altova XSLT 1.0-Prozessors, des Altova XSLT 2.0-Prozessors und des Altova XQuery-Prozessors.

© 2012 Altova GmbH

3.1 Altova XML Validator

Der Altova XML Validator implementiert die folgenden Regeln und entspricht diesen:

- XML 1.0 (Fourth Edition)
- XML Namespaces (1.0)
- XML Schemas (Structures)
- XML Schema (Datatypes)

3.2 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 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 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()` 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>
```

 © 2012 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</i>.</para> or
<para>This is <b>bold&#x20;</b> <i>italic</i>.</para> or
<para>This is <b>bold</b><i>&#x20;italic</i>.</para>
```

Wenn eines der oben genannten `para`-Elemente mit demselben XSLT Template verarbeitet wird, erhalten Sie folgendes Ergebnis:

```
This is bold italic.
```

3.3 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.

© 2012 Altova GmbH

3.3.1 Allgemeine Informationen

Der Altova XSLT 2.0-Prozessor entspricht der XSLT 2.0 Recommendation vom 23 January 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"
   ...
</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-Konstruktoren 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 January 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.

© 2012 Altova GmbH

3.3.2 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.

xsl:result-document
Zusätzlich werden die folgenden Kodierungen unterstützt: `x-base16tobinary` und `x-base64tobinary`.

function-available
Die Funktion überprüft, ob in-scope-Funktionen (XSLT 2.0, XPath 2.0 und Erweiterungsfunktionen) 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. Zusätzlich werden die folgenden Kodierungen unterstützt: `binarytobase16` und `binarytobase64`.

unparsed-text-available
Das Attribut `href` akzeptiert (i) relative Pfade für Dateien im Basis-URI-Ordner und (ii) absolute Pfade mit oder ohne das `file://`-Protokoll. Zusätzlich werden die Kodierungen `x-binarytobase16` und `x-binarytobase64` unterstützt.

Anmerkung: Die folgenden Kodierungswerte, die in früheren Versionen von AltovaXML implementiert waren, werden nun nicht mehr verwendet: `base16tobinary`, `base64tobinary`, `binarytobase16` und `binarytobase64`.

3.4 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 January 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 January 2007 aus dem XPath Datentypen-Namespace in den XML-Schema Namespace verschoben

 © 2012 Altova GmbH

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:

```
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.

 © 2012 Altova GmbH

3.5 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 January 2007.

3.5.1 Allgemeine Informationen

Standardkonformität

- Der Altova XPath 2.0-Prozessor implementiert die XPath 2.0 Recommendation vom 23 January 2007. Der Altova XQuery 1.0-Prozessor implementiert die XQuery 1.0 Recommendation vom 23 January 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 January 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 (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

© 2012 Altova GmbH

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.

3.5.2 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`	• Das Argument ist eine relative URI, die gegen die aktuelle Basis-URI aufgelöst wird.
	• Wenn mit der aufgelösten URI eine XML-Datei identifiziert wird, so wird diese XML-Datei als Katalog behandelt, der eine Sammlung von Dateien referenziert. Diese Datei muss die folgende Form haben:
	`<collection>`
	`  <doc href="uri-1" />`
	`  <doc href="uri-2" />`
	`  <doc href="uri-3" />`
	`</collection>`
	Die von den `href` Attributen referenzierten Dateien werden geladen und ihre Dokument-Nodes werden als Sequenz zurückgegeben.
	• Wenn mit der aufgelösten URI keine XML-Datei mit der oben beschriebenen Katalogstruktur identifiziert wird, dann wird der Argument-String (in dem die Platzhalterzeichen ? und * zulässig sind) als Suchstring verwendet. XML-Dateien, deren Namen dem Suchausdruck entsprechen, werden geladen und ihre Dokument-Nodes werden als Sequenz zurückgegeben. Siehe Beispiel unten.
	• XSLT-Beispiel: Der Ausdruck `collection("c:\MyDocs\*.xml")//Title` gibt eine Sequenz aller `DocTitle` Elemente in den `.xml` Dateien im Ordner `MyDocs` zurück.
	• XQuery-Beispiel: Der Ausdruck `{for $i in collection(c:\MyDocs\*.xml) return element doc{base-uri($i)}}` gibt die Basis-URIs aller `.xml` Dateien im

 © 2012 Altova GmbH

	Ordner `MyDocs` zurück, wobei jede URI sich innerhalb eines `doc` Elemente befindet. • Die Standard-Collection ist leer.
`count`	• Siehe Anmerkungen zu Whitespaces im Abschnitt Allgemeine Informationen.
`current-date, current-dateTime, current-time`	• Das aktuelle Datum und die Uhrzeit werden von der Systemuhr abgerufen. • Die Zeitzone stammt aus der vom Auswertungskontext gelieferten impliziten Zeitzone; die implizite Zeitzone wird von der Systemuhr abgerufen. • Die Zeitzone wird immer im Ergebnis definiert.
`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.
`lower-case`	• Es wird der Unicode-Zeichensatz unterstützt.
`normalize-unicode`	• Es werden die Normalisierungsformulare NFC, NFD, NFKC und NFKD unterstützt.
`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`	• Der Unicode-Zeichensatz wird unterstützt.

© 2012 Altova GmbH

3.6 Erweiterungen

Es gibt in Programmiersprachen wie Java und C# eine Reihe von fertigen Funktionen, die nicht als XPath 2.0 / XQuery 1.0-Funktionen oder XSLT 2.0-Funktionen zur Verfügung stehen. Ein gutes Beispiel für solche Funktionen sind die mathematischen in Java verfügbaren Funktionen wie z.B. `sin()` und `cos()`. Stünden diese Funktionen für die Erstellung von XSLT Stylesheets und XQuery-Abfragen zur Verfügung, würde sich der Einsatzbereich von Stylesheets und Abfragen erweitern und die Erstellung von Stylesheets wäre viel einfacher.

Die Altova-Prozessoren (XSLT 1.0, XSLT 2.0 und XQuery 1.0), die in einer Reihe von Altova-Produkten verwendet werden, unterstützen die Verwendung von Erweiterungsfunktionen in Java und .NET. Zusätzlich unterstützen die Altova XSLT-Prozessoren MSXSL Scripts für XSLT 1.0 und 2.0.

Beachten Sie, dass Erweiterungsfunktionen immer von XPath-Ausdrücken aus aufgerufen werden. In diesem Abschnitt wird beschrieben, wie Sie Erweiterungsfunktionen und MSXSL Scripts in Ihren XSLT Stylesheets und XQuery-Abfragen verwenden können. Diese Beschreibungen finden Sie in den folgenden Abschnitten:

- Java-Erweiterungsfunktionen
- .NET-Erweiterungsfunktionen
- MSXSL Scripts für XSLT
- Altova Erweiterungsfunktionen

Hauptsächlich werden dabei die folgenden beiden Punkte behandelt: (i) Wie Funktionen in den entsprechenden Bibliotheken aufgerufen werden; und (ii) welche Regeln beim Konvertieren von Argumenten in einem Funktionsaufruf in das erforderliche Format der Funktion befolgt werden und welche Regeln bei der Rückwärtskonvertierung (Funktionsresultat in XSLT/XQuery Datenobjekt) befolgt werden.

Voraussetzungen
Damit die Erweiterungsfunktionen unterstützt werden, muss auf dem Rechner, auf dem die XSLT-Transformation oder die XQuery-Ausführung stattfindet, eine Java Runtime-Umgebung (zum Aufrufen der Java-Funktionen) installiert sein oder es muss Zugriff auf eine solche bestehen.

3.6.1 Altova Erweiterungsfunktionen

Altova Erweiterungsfunktionen befinden sich im Namespace **http://www.altova.com/xslt-extension** und werden in diesem Abschnitt mit dem Präfix **altova:** gekennzeichnet, das an den oben angegebenen Namespace gebunden ist.

In der aktuellen Version Ihres Altova-Produkts werden die folgenden Erweiterungsfunktionen auf die unten beschriebene Art unterstützt.

Allgemeine Funktionen

XPath-Funktionen
Die folgenden Funktionen können im Zusammenhang mit XPath verwendet werden:

- **altova:generate-auto-number()**
- **altova:reset-auto-number()**
- **altova:get-temp-folder()**

XSLT-Funktionen:
Diese Funktionen können genau wie die XSLT 2.0 Funktionen `current-group()` oder `key()` im Zusammenhang mit XSLT verwendet werden:

- **altova:evaluate()**
- **altova:distinct-nodes()**
- **altova:encode-for-rtf()**
- **altova:xbrl-labels()**
- **altova:xbrl-footnotes()**

Allgemeine Funktionen

In der aktuellen Version Ihres Altova-Produkts werden die folgenden Erweiterungsfunktionen auf die unten beschriebene Art unterstützt. Beachten Sie jedoch, dass eine oder mehrere dieser Funktionen in zukünftigen Produktversionen eventuell nicht mehr unterstützt werden, bzw. dass sich das Verhalten einzelner Funktionen ändern kann. Um Informationen über die Unterstützung für Altova Erweiterungsfunktionen in der jeweiligen Release zu erhalten, schlagen Sie bitte in der Dokumentation der jeweils aktuellen Release nach.

XPath- und XQuery-Funktionen
Die folgenden Funktionen können im Zusammenhang mit XPath verwendet werden:

- **altova:generate-auto-number()**
- **altova:reset-auto-number()**
- **altova:get-temp-folder()**

XSLT-Funktionen:
Diese Funktionen können genau wie die XSLT 2.0 Funktionen `current-group()` oder `key()` im Zusammenhang mit XSLT verwendet werden:

- **altova:evaluate()**
- **altova:distinct-nodes()**
- **altova:encode-for-rtf()**
- **altova:xbrl-labels()**
- **altova:xbrl-footnotes()**

XPath-Funktionen
Diese Funktionen können im Zusammenhang mit XPath verwendet werden:

© 2012 Altova GmbH

altova:generate-auto-number(id als xs:string, start-with als xs:integer, increment als xs:integer, reset-on-change als xs:string)
Generiert eine Reihe von Zahlen mit der definierten ID. Die Anfangszahl (Ganzzahl) und die Inkrementierung werden definiert.

altova:reset-auto-number(id als xs:string)
Diese Funktion setzt die automatische Nummerierung der mit dem ID-Argument definierten automatischen Nummerierung zurück. Die Reihe wird auf den Anfangsganzzahlwert der Reihen zurückgesetzt (siehe `altova:generate-auto-number` oben).

altova:get-temp-folder als xs:string
Ruft den temp-Ordner ab.

XSLT-Funktionen

Diese Funktionen können genau wie die XSLT 2.0 Funktionen `current-group()` oder `key()` im Zusammenhang mit XSLT verwendet werden:

altova:evaluate()
Die `altova:evaluate()` Funktion erhält einen XPath-Ausdruck als obligatorisches Argument, der als String übergeben wird, und gibt das Resultat des ausgewerteten Ausdrucks zurück.

```
altova:evaluate(XPathExp as xs:string)
```

Beispiel:

```
altova:evaluate('//Name[1]')
```

Beachten Sie im obigen Beispiel, dass der Ausdruck `//Name[1]` durch Einschließen in einfache Anführungszeichen als String übergeben wird. Die Funktion `altova:evaluate` gibt den Inhalt des ersten `Name`-Elements im Dokument zurück.

Die Funktion `altova:evaluate` kann zusätzliche (optionale) Argumente erhalten. Diese Argumente sind die Werte der einzelnen Variablen und haben die Namen `p1`, `p2`, `p3`... `pN`, die im XPath-Ausdruck verwendet werden können.

```
altova:evaluate(XPathExp as xs:string [, p1value ... pNvalue])
```

wobei

- die Variablennamen die Form `pX` haben müssen, wobei `X` eine Ganzzahl ist
- die Reihenfolge der Argumente der Funktion vom zweiten Argument an der Reihenfolge der Variablen mit den Namen `p1` bis `pN` entspricht. Daher wird das zweite Argument der Wert der Variablen `p1`, das dritte Argument der der Variablen `p2`, usw.
- die Werte der Variablen den Typ `item*` haben müssen

Beispiel:

```
<xsl:variable name="xpath" select="'$p3, $p2, $p1'" />
<xsl:value-of select="altova:evaluate( $xpath, 10, 20, 'hi' )" />
```

Gibt "`hi 20 10`" zurück

Beachten Sie im obigen Beispiel folgende Punkte:

- Das zweite Argument des Ausdrucks `altova:evaluate` ist der der Variablen `$p1` zugewiesene Wert, das dritte Argument ist das der Variablen `$p2` zugewiesene usw.
- Beachten Sie, dass das vierte Argument der Funktion ein String-Wert ist. Als String-Wert wird dieser innerhalb von Anführungszeichen gesetzt.
- Das `select` Attribut des Elements `xs:variable` liefert den XPath-Ausdruck. Da dieser Ausdruck den Typ `xs:string`, haben muss, wird er in einfache Anführungszeichen gesetzt..

Im Folgenden ein weiteres Beispiel für die Verwendung:

```
<xsl:variable name="xpath" select="'$p1'" />
<xsl:value-of select="altova:evaluate( $xpath, //Name[1] )" />
```
Gibt den Wert des ersten `Name` *Elements aus.*

```
<xsl:variable name="xpath" select="'$p1'" />
<xsl:value-of select="altova:evaluate( $xpath, '//Name[1]' )" />
```
Gibt "`//Name[1]`*" aus*

Die `altova:evaluate()` Erweiterungsfunktion ist in Situationen nützlich, in denen ein XPath-Ausdruck im XSLT-Stylesheet einen oder mehrere Teile enthält, die dynamisch ausgewertet werden müssen. Angenommen ein Benutzer gibt seinen Request für das Sortierkriterium ein und das Sortierkriterium ist im Attribut `UserReq/@sortkey`. gespeichert. Im Stylesheet könnten Sie den folgenden Ausdruck haben:

```
<xsl:sort select="altova:evaluate(../UserReq/@sortkey)" order="ascending"/
>
```

Die `altova:evaluate()` Funktion liest das `sortkey` Attribut des `UserReq` Child-Elements des Parent des Kontext-Node. Angenommen der Wert des `sortkey` Attributs ist `Price`, dann wird von der `altova:evaluate()` Funktion `Price` zurückgegeben und wird zum Wert des `select` Attributs:

```
<xsl:sort select="Price" order="ascending"/>
```

Wenn diese `sort` Anweisung im Kontext eines Elements namens `Order` vorkommt, dann werden die `Order` Elemente nach den Werten Ihrer `Price` Children sortiert. Alternativ dazu, wenn der Wert von `@sortkey` z.B. `Date` ist, werden die `Order` Elemente nach den Werten ihrer `Date` Children sortiert. Das Sortierkriterium für `Order` wird also zur Laufzeit aus dem `sortkey` Attribut ausgewählt. Diese hätte man mit einem Ausdruck wie dem folgenden nicht bewerkstelligen können:

```
<xsl:sort select="../UserReq/@sortkey" order="ascending"/>
```

Im oben gezeigten Beispiel wäre das Sortierkriterium das `sortkey` Attribut selbst, nicht `Price` oder `Date` (oder jeder beliebige andere Inhalt von `sortkey`)

Variablen können in der Erweiterungsfunktion `altova:evaluate()` wie im Beispiel unten gezeigt verwendet werden:

- Statische Variablen: `<xsl:value-of select="$i3, $i2, $i1" />`
  Gibt die Werte von drei Variablen aus.

- Dynamischer XPath-Ausdruck mit dynamischen Variablen:
  ```
  <xsl:variable name="xpath" select="'$p3, $p2, $p1'" />
  <xsl:value-of select="altova:evaluate( $xpath, 10, 20, 30 )" />
  ```
  Gibt "`30 20 10`*" aus*

- Dynamischer XPath-Ausdruck ohne dynamische Variable:
  ```
  <xsl:variable name="xpath" select="'$p3, $p2, $p1'" />
  <xsl:value-of select="altova:evaluate( $xpath )" />
  ```

 © 2012 Altova GmbH

Gibt einen Fehler aus.: Es wurde keine Variable für `$p3` *definiert*

Hinweis:
Der statische Kontext enthält Namespaces, Typen und Funktionen - aber keine Variablen - aus der aufrufenden Umgebung. Die Basis-URI und der Standard-Namespace werden vererbt.

`altova:distinct-nodes()`
Die `altova:distinct-nodes()` Funktion erhält eine Gruppe von einem oder mehreren Nodes als Input und gibt dieselbe Gruppe ohne Nodes mit doppelt vorhandenen Werten zurück. Der Vergleich wird mittels der XPath/XQuery-Funktion `fn:deep-equal` durchgeführt.

```
altova:distinct-nodes( $arg as node()* )  as node()*
```

`altova:encode-for-rtf()`
Die `altova:encode-for-rtf()` Funktion konvertiert den Input-String in Code für RTF.

```
altova:encode-for-rtf( $inputstr as xs:string?,
$preserveallwhitespace as xs:boolean,
$preservenewlines as xs:boolean) as xs:string
```

Whitespaces und neue Zeilen werden gemäß dem für die entsprechenden Parameter definierten Booleschen Wert beibehalten.

`altova:xbrl-labels()`
Die `altova:xbrl-labels()` Funktion erhält zwei Input-Argumente: einen Node-Namen und den Pfad der Taxonomiedatei, die den Node enthält. Die Funktion gibt die XBRL Labels zurück, die mit dem Input-Node verknüpft sind.

```
altova:xbrl-labels( $name as xs:QName, $file as xs:string )  as node()*
```

`altova:xbrl-footnotes()`
Die `altova:footnotes()` Funktion erhält einen Node als Input-Argument und gibt die durch den Input-Node referenzierte Gruppe der XBRL-Fußnoten-Nodes zurück.

```
altova:footnotes( $arg as node() )  as node()*
```

Kapitel 4

Lizenzinformationen

4 Lizenzinformationen

Dieser Abschnitt enthält die folgenden Informationen:

- Informationen über den Vertrieb dieses Software-Produkts
- Informationen im Zusammenhang mit dem intellektuellen Eigentum dieses Software-Produkts
- 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.

© 2012 Altova GmbH

4.1 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 Altova Developer Lizenz 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.

Weitergabe der Software an andere Mitarbeiter in Ihrem Unternehmen zu Testzwecken
Wenn Sie die Evaluierungsversion der Software auch anderen Personen in Ihrem Unternehmen über das Netzwerk zur Verfügung stellen möchten oder wenn Sie sie auf einem PC installieren möchten, der nicht mit dem Internet verbunden ist, sollten Sie nur das Installationsprogramm weitergeben - vorausgesetzt es wurde nicht modifiziert. Jeder, der das von Ihnen zur Verfügung gestellte Installationsprogramm aufruft, muss einen eigenen Evaluierungs-Keycode für 30 Tage anfordern. Nach Ablauf des Testzeitraums, muss eine Lizenz erworben werden, damit das Produkt weiter verwendet werden kann.

Nähere Informationen finden Sie in der Altova Developer Lizenzvereinbarung am Ende dieses Handbuchs.

4.2 Rechte am geistigen Eigentum

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. Altova behält sich 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. Mitteilungen über geltend gemachte Verstöße gegen geistige Eigentumsrechte sind an den Copyright Agenten von Altova zu richten (nähere Angaben dazu finden Sie auf der Website von Altova).

Altova Software enthält Software-Komponenten Dritter, die ebenfalls urheberrechtlich geschützt sind. Unter anderem sind diese Urheberrechte ausführlich unter http://www.altova.com/legal_3rdparty.html aufgelistet.

Alle anderen Namen oder Warenzeichen sind das Eigentum ihrer jeweiligen Inhaber.

© 2012 Altova GmbH

4.3 Lizenzvereinbarung

DIES IST EIN RECHTSGÜLTIGES DOKUMENT -- BITTE BEWAHREN SIE ES SORGFÄLTIG AUF

ALTOVA DEVELOPER LIZENZVERTRAG

für die AltovaXML® Reporting Edition Software
und die AltovaXML® Community Edition Software

Lizenzgeber:

Altova GmbH
Rudolfsplatz 13a/9
A-1010 Wien
Österreich

Wichtig - Bitte sorgfältig lesen. Benutzerinformation:

Dieser Altova Developer Lizenzvertrag für AltovaXML® ("DLAXML") regelt Ihr Recht die (i) AltovaXML Reporting Edition Software ("AXRE") und die (ii) AltovaXML Community Edition Software ("AXCE") zu verwenden, zu reproduzieren, zu bündeln, zu integrieren und zu vertreiben, (im Folgenden einzeln oder gesamt als „die AltovaXML Software" bezeichnet). Ihre Rechte aus dieser Lizenzvereinbarung hängen von der spezifischen Software Edition ab, über die Sie eine Lizenz erworben haben, da auf die Editionen verschiedene Rechte und Einschränkungen anwendbar sind, wie im einzelnen im Folgenden festgehalten wird. Dieser Developer-Lizenzvertrag („Lizenzvertrag") ist ein rechtsgültiger Vertrag, der zwischen Ihnen und Altova GmbH („Altova") abgeschlossen wurde. Bevor Sie die von Altova bereitgestellte 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 AltovaXML Software installieren, die AltovaXML Software inkludieren, vertreiben 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. 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 oder zum Vertrieb der AltovaXML Software berechtigt und müssen alle heruntergeladenen Kopien der AltovaXML Software, die sich in Ihrem Besitz oder unter Ihrer Kontrolle befinden, vernichten. Bitte besuchen Sie unsere Website unter http://www.altova.com/de/ALTOVADLAXML, um eine Kopie dieses Lizenzvertrags herunterzuladen und auszudrucken. Die Datenschutzbestimmungen können unter http://www.altova.com/de/privacy eingesehen werden.

1. AltovaXML Reporting Edition ("AXRE") Software Bestimmungen

(a) Lizenzgewährung. Sofern Sie diesem Lizenzvertrag zugestimmt haben, gewährt Ihnen Altova eine nicht ausschließliche und nicht übertragbare Lizenz (mit Ausnahme der nachfolgenden Bestimmungen), ohne das Recht zur Vergabe von Sublizenzen,

(i) zur Installation und Verwendung eines Exemplars der AXRE Software auf einem oder mehreren kompatiblen Einzelplatzrechnern, Workstations oder Servern sowie zur Entwicklung Ihrer eigenen Software-Applikationen, die die AXRE Software und Dokumentation

beinhalten, wobei die maximale Zahl an Computern, die im Lizenzumfang enthalten sind, nicht überschritten werden darf. Sie dürfen ein Exemplar der AXRE Software auf einem Dateiserver innerhalb Ihres internen Netzwerks zum Zweck des Herunterladens und Installieren der AXRE Software auf anderen Computern innerhalb Ihres Netzwerks bis zur maximalen Anzahl von betrieblich genutzten Computern installieren.

(ii) Sie dürfen auf Ihrem Netzwerk den Zugriff auf einen Webservice oder einen anderen von Ihnen entwickelten Dienst einräumen, der die Funktionen der AXRE Software in einer von Ihnen entwickelten Applikation nutzt und der einen Bericht oder andere Ergebnisdaten anzeigt, die von Ihrem Dienst in Antwort auf die und auf Basis von den Daten, die von einem diesen Dienst aufrufenden Benutzer gesendet wurden, generiert werden.

(iii) Die Reproduktion und Weitergabe der AXRE Software und/oder Dokumentation in ausführbarer Form auf elektronischem Weg, durch Herunterladen von Ihrer Website oder über physische Medien an Dritte in Verbindung mit einer von Ihnen entwickelten Software-Applikation, die die AXRE Software enthält, ist gestattet und unterliegt den unten angeführten Einschränkungen und Voraussetzung für die Übertragung.

(iv) Sie dürfen eine Kopie der AXRE 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 AXRE Software gemäß diesem Lizenzvertrag erfolgt.

(b) Einschränkungen und Voraussetzungen für die Übertragung. Zusätzlich zu den in anderen Abschnitten dieses Lizenzvertrags festgelegten Einschränkungen und Bestimmungen unterliegt Ihre Lizenz zur Weitergabe der AXRE Software und/oder Dokumentation den folgenden Einschränkungen: **(i)** Sie dürfen alle Ihre Rechte zur Verwendung der AXRE Software-Komponente Ihrer Software-Applikation und/oder der Dokumentation an eine andere natürliche oder juristische Person unter der Voraussetzung übertragen, dass alle Endbenutzer eine Lizenz zur Verwendung der AXRE Software-Komponente Ihrer Software Applikation von Altova oder einem autorisierten Vertriebshändler erwerben; **(ii)** Sie dürfen die AXRE Software und/oder Dokumentation nicht als Stand-alone-Produkt, sondern nur als Komponente Ihrer Software-Applikation bzw. in Verbindung mit Ihrer Software-Applikation zur Verfügung stellen; **(iii)** Sie müssen das von Altova zur Verfügung gestellte Installationsprogramm für AXRE in unveränderter Form verwenden und dürfen diesen DLAXML-Lizenzvertrag (der bei der Installation angezeigt wird und den ein Endbenutzer akzeptieren muss, um die AXRE Software installieren oder verwenden zu können) oder andere Dateien nicht ändern oder entfernen;; und **(iv)** andere Altova-Produkte dürfen im Rahmen dieses Lizenzvertrags nicht vertrieben oder verwendet werden.

(c) Freischaltcodes, Upgrades und Updates. Wenn Sie sich entschließen die AXRE Software Lizenzen entweder von der Altova GmbH oder einem autorisierten Vertriebshändler zu erwerben, erhalten Sie einen Kauf Freischaltcode. Es ist Ihnen untersagt den Freischalt Code für die Software zu re-lizenzieren, zu reproduzieren, oder weiterzugeben, außer mit der ausdrücklichen schriftlichen Erlaubnis von Altova. Wenn die von Ihnen in Lizenz erworbene Software ein Upgrade oder Update ist, ersetzt das Upgrade oder Update das sie zuletzt heruntergeladen oder installiert haben, die zuvor lizenzierte Kopie der AXRE Software. Durch das betreffende Upgrade oder Update und die dazugehörigen Freischaltcodes wird keine zweite Lizenz für die Software gewährt, und Sie dürfen das Upgrade oder Update nicht zusätzlich zu dem Software-Exemplar verwenden, das dadurch ersetzt wird und dessen Lizenz abgelaufen ist.

(d) Support und Wartung. Altova bietet Support und Wartungs-Pakete (“SMP”) für die

AXRE Software an, für die Sie eine Lizenz erworben haben. 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.

(i) 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 AXRE Software bestehende Maß hinausgehen, beinhaltet. In jedem Fall erhalten Sie dreißig (30) Tage lang ab Kauf der Software (der „Supportzeitraum" im Sinne dieses Absatzes 1(d)) 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.

(ii) 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 AXRE 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.

(iii) 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 AXRE 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 AXRE 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 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 nach dem Ermessen von Altova nicht wesentlich beeinträchtigen, zu beheben Sie erhalten von Altova unter keinen Umständen Beratung, Unterstützung oder Hilfestellung, die allgemeiner Natur ist und nicht in unmittelbarem Zusammenhang mit der AXRE Software steht.

(iv) Für ein Update der AXRE Software kann es notwendig sein, Software, welche nicht von diesem Lizenzvertrag abgedeckt wird, vor der Installation zu aktualisieren (updaten).

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 festgelegten Verpflichtungen durch Altova versteht sich unter der Bedingung, dass Sie die AXRE 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 AXRE Software auf einen der folgenden Gründe zurückzuführen ist: (a) Fehler, die durch die Verlegung der Software auf einen anderen Rechner oder Speicherort hervorgerufen werden; (b) Änderungen, Modifikationen oder Versuche, die Software abzuwandeln, die von Altova nicht schriftlich genehmigt wurden; (c) äußere Einflüsse auf die Software, wie z.B. Naturkatastrophen, Stromausfälle, Stromschwankungen oder Computerausfälle; (d) Ihr Versäumnis die Software auf dem von Altova festgelegten Release Level zu halten; oder (e) die Nutzung der Software ohne vorherige Genehmigung durch Altova zusammen mit einer anderen Software. Sie alleine sind dafür verantwortlich: (a) allen Betriebs- und Fehlerbehebungsanleitungen von Altova Folge zu leisten, Altova unverzüglich über Fehler oder Defekte an der AXRE Software zu informieren und Altova eine genaue Beschreibung dieser Fehler und/oder Defekte zu liefern; (b) für den Schutz Ihrer vertraulichen Informationen zu sorgen; und (c) Datensicherungssysteme und -abläufe für die Wiederherstellung verlorener oder geänderter Dateien, Daten oder Programme einzurichten und anzuwenden.

(e) Eingeschränkte Garantie. Altova garantiert der Person/Rechtspersönlichkeit, die ursprünglich die Lizenz für die Verwendung der AXRE 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 1(d) 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 AXRE 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 AXRE 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 AXRE 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. Diese eingeschränkte Garantie gilt nicht für Test-Software. 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

 © 2012 Altova GmbH

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 JA NACH LAND/RECHTSORDNUNG UNTERSCHIEDLICH SEIN KÖNNEN.

(f) Haftungsbeschränkung und Ansprüche in Zusammenhang mit Urheberrechtsverletzungen. 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 AXRE 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 AXRE-SOFTWAREPRODUKT ENTRICHTETEN BETRAGES BESCHRÄNKT. Da es in einigen Ländern und Rechtsordnungen nicht zulässig ist, die Haftung für Schäden auszuschließen oder zu 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. 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 AXRE 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 1(f) 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 AXRE 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 AXRE 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 AXRE 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.

(g) Anwendbare Lizenzvertragsbestimmungen. Die in den Abschnitten 1 und 3 festgelegten Bestimmungen gelten für die AXRE Software.

2. AltovaXML Community Edition ("AXCE") Software Bestimmungen

(a) Lizenzgewährung. Sofern Sie diesem Lizenzvertrag zugestimmt haben, gewährt Ihnen Altova eine (mit Ausnahme der unten angeführten Bestimmungen) nicht ausschließliche und nicht übertragbare eingeschränkte Lizenz, ohne das Recht Sublizenzen einzuräumen**,** zur Entwicklung von Software-Applikationen, die die ACXE Software und/oder Dokumentation beinhalten, zur Reproduktion der AXCE Software und/oder Dokumentation sowie zur Weitergabe der AXCE Software an Dritte in ausführbarer Form in Verbindung mit einer von Ihnen entwickelten Applikation, die die AXCE Software enthält, auf elektronischem Weg, durch Herunterladen von Ihrer Website oder über ein physisches Medium gemäß den unten angeführten Einschränkungen und Voraussetzungen für die Übertragung. Sie dürfen die AXCE Software zum Zweck des Herunterladens und Installierens der AXCE Software (auf einer beliebigen Anzahl von Client-Rechnern innerhalb Ihres Netzwerks) auf einem Server innerhalb Ihres Netzwerks installieren.

(b) Einschränkungen und Voraussetzungen für die Übertragung. Zusätzlich zu den in anderen Abschnitten dieses Lizenzvertrags festgelegten Einschränkungen und Bestimmungen unterliegt Ihre Lizenz zur Weitergabe der AXCE Software und/oder Dokumentation den folgenden Einschränkungen: **(i)** Die AXCE Software kann nur lizenziert und darf nicht verkauft werden; **(ii)** es ist Ihnen nicht gestattet die AXCE Software und/oder Dokumentation als Stand-Alone Produkt zugänglich zu machen und, wenn sie als Teil eines Produkt-Bundles vertrieben wird, ist es Ihnen erlaubt hiefür einen Preis festzusetzen, sofern dieses Produkt-Bundle zu demselben oder einem niedrigeren Preis als jenem, zu dem ein gleichwertiges Produkt-Bundle, welches die AXCE Software nicht enthält, lizenziert wird; **(iii)** Sie müssen das von Altova zur Verfügung gestellte Setup Programm für die AXCE Software unverändert benutzen und dürfen Altovas Lizenzvertrag (welcher während des Installationsprozesses aufscheint und vom Endbenutzer akzeptiert werden muss um die AXCE Software installieren oder benutzen zu können) oder andere Dateien nicht beschädigen, verändern oder entfernen; und **(iv)** andere Altova-Produkte dürfen im Rahmen dieses Lizenzvertrags nicht vertrieben oder verwendet werden.

(c) Garantie, Haftungseinschränkungen. DIE AXCE SOFTWARE WIRD IHNEN GRATIS UND **WIE BESEHEN** („AS-IS") ZUR VERFÜGUNG GESTELLT. ALTOVA GIBT KEINERLEI GEWÄHRLEISTUNG FÜR DIE AXCE SOFTWARE AB. ALTOVA UND IHRE ZULIEFERER GEBEN SOWOHL AUSDRÜCKLICH ALS AUCH STILLSCHWEIGEND UND AUF JEDE ANDERE WEISE IM GESETZLICH GRÖßTMÖGLICHEN AUSMAß KEINERLEI GEWÄHRLEISTUNGEN ODER ZUSICHERUNGEN, INSBESONDERE JEDOCH NICHT AUSSCHLIEßLICH, FÜR ALLE IMPLIZITEN GEWÄHRLEISTUNGEN DER EIGNUNG FÜR EINEN BESTIMMTEN ZWECK, FÜR MARKTGÄNGIGKEIT, FÜR BEFRIEDIGENDE QUALITÄT, FÜR INFORMATIONSGEHALT ODER RICHTIGKEIT, FÜR UNGESTÖRTE NUTZUNG, EIGENTUMSRECHTE UND DIE NICHTVERLETZUNG VON RECHTEN DRITTER, AB. ALTOVA ÜBERNIMMT KEINE GEWÄHRLEISTUNG FÜR DIE FEHLERFREIHEIT DER AXCE SOFTWARE ODER, DASS DIESE OHNE

UNTERBRECHUNGEN LAUFFÄHIG SIND. FALLS NACH ZWINGEND ANWENDBAREM RECHT IRGENDWELCHE GEWÄHRLEISTUNGEN IN BEZUG AUF DIE AXCE SOFTWARE VORGESEHEN SIND, WERDEN SOLCHEN GEWÄHRLEISTUNGEN FÜR EINEN ZEITRAUM VON DREISSIG (30) TAGEN AB DER INSTALLATION ODER DEM BEGINN DER VERWENDUNG DER SOFTWARE, JE NACHDEM WAS FRÜHER IST, EINGESCHRÄNKT. IN MANCHEN STAATEN ODER LÄNDERN IST DER AUSSCHLUSS VON IMPLIZITEN GEWÄHRLEISTUNGEN NICHT ERLAUBT, DAHER KÖNNTE DER OBEN ANGEFÜHRTE GEWÄHRLEISTUNGSAUSSCHLUSS SIE NICHT BETREFFEN. DIESE GEWÄHRLEISTUNG GIBT IHNEN SPEZIFISCHE RECHTE. SIE KÖNNTEN AUCH ANDERE RECHTE BESITZEN, WELCHE VON STAAT ZU STAAT UND VON LAND ZU LAND UNTERSCHIEDLICH SIND. SIE TRAGEN DIE ALLEINIGE VERANTWORTUNG FÜR DIE EIGNUNG UND ZWECKMÄSSIGKEIT DER ACDE ODER ACBE SOFTWARE FÜR DEN BEABSICHTIGTEN VERWENDUNGSZWECK UND WERDEN ALTOVA FÜR JEGLICHE ANSPRÜCHE UND FORDERUNGEN DRITTER, DIE AUF DIE EIGNUNG UND ZWECKMÄSSIGKEIT DER VON IHNEN VERWENDETEN ACDE ODER ACBE SOFTWARE BEGRÜNDET SIND, SCHAD- UND KLAGLOS HALTEN.

(d) Haftungseinschränkung. SOWEIT DIE ANWENDBAREN GESETZE DIES IM GRÖßTMÖGLICHEN AUSMAß ZULASSEN, HAFTEN ALTOVA ODER IHRE ZULIEFERER, IN KEINEM FALL FÜR KONKRETE, ZUFÄLLIG ENTSTANDENE, MITTELBARE, UNMITTELBARE ODER FOLGESCHÄDEN WELCHER ART AUCH IMMER (INSBESONDERE SCHÄDEN AUS ENTGANGENEM GEWINN, GESCHÄFTSUNTERBRECHUNGEN, VERLUST VON GESCHÄFTSINFORMATIONEN ODER ANDEREN FINANZIELLEN VERLUSTEN), DIE DURCH DIE NUTZUNG ODER DIE UNMÖGLICHKEIT DER NUTZUNG DER AUTHENTIC SOFTWARE ODER DIE NICHTERBRINGUNG VON SUPPORTLEISTUNGEN ENTSTANDEN SIND, ODER DIESES LIZENZVERTRAGES, SELBST WENN ALTOVA AUF DIE MÖGLICHKEIT EINES SOLCHEN SCHADENS AUFMERKSAM GEMACHT WURDE. SOFERN UND SOWEIT KEIN HAFTUNGSSAUSSCHLUSS, SONDERN NUR EINE HAFTUNGSBESCHRÄNKUNG GESETZLICH ZULÄSSIG IST, BESCHRÄNKT SICH DIE HAFTUNG VON ALTOVA UND DESSEN LIEFERANTEN AUF INSGESAMT FÜNF US-DOLLAR (5 USD). DA IN MANCHEN STAATEN UND JURISDIKTIONEN DER AUSSCHLUSS ODER DIE BEGRENZUNG VON HAFTUNGEN NICHT MÖGLICH IST, IST DIE OBEN ANGEFÜHRTE HAFTUNGSBESCHRÄNKUNG AUF SIE VIELLEICHT NICHT ANWENDBAR. IN SOLCHEN STAATEN UND JURISDIKTIONEN WIRD DIE HAFTUNG VON ALTOVA UND IHREN ZULIEFERERN IM GRÖSSTMÖGLICHEN RECHTLICH ERLAUBTEN AUSMASS EINGESCHRÄNKT. DIE BISHER ANGEFÜHRTEN EINSCHRÄNKUNGEN DER HAFTUNG SIND ANZUWENDEN AUF DIE OBEN ANGEFÜHRTEN GEWÄHRLEISTUNGEN; GEWÄHRLEISTUNGSAUSSCHLÜSSE UND ALLE ANDEREN ASPEKTE DIESES LIZENZVERTRAGES.

(e) Support. Altova ist nicht verpflichtet technischen Support für die AXCE Software zur Verfügung zu stellen.

(f) Anwendbare Lizenzvertragsbestimmungen. Die in den Abschnitten 2 und 3 festgelegten Bestimmungen gelten für die AXCE Software.

3. AltovaXML Software (AXRE und AXCE) Software Bestimmungen

Die in Abschnitt 3 angeführten Bestimmungen gelten für die AXRE- und die AXCE-Software-Lizenz und ergänzen die speziellen Bestimmungen, die für diese Software-Lizenzen gelten.

(a) Eigentumsrecht. Dieser Lizenzvertrag räumt Ihnen ein eingeschränktes Recht darauf ein, die AltovaXML Software und/oder -Dokumentation wie oben festgelegt zu reproduzieren und zu vertreiben. Das Eigentum an allen Kopien der AltovaXML Software und Dokumentation sowie alle Urheberrechte und Rechte am geistigen Eigentum stehen Altova und seinen Lieferanten zu. Das Eigentumsrecht an der AltovaXML Software wird Ihnen nicht übertragen.

Alle Exemplare der AltovaXML Software und alle von Ihnen angefertigte Kopien verbleiben - vorbehaltlich der Ihnen durch diesen Lizenzvertrag eingeräumten Rechte zur Verwendung und zum Vertrieb - im Eigentum von Altova. Alle Rechte, welche Ihnen nicht durch diesen Lizenzvertrag ausdrücklich gewährt werden, sind Altova vorbehalten.

(b) Anerkennung der Rechte von Altova. Die AltovaXML 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 AltovaXML Software stellen wertvolle Betriebsgeheimnisse und vertrauliche Informationen von Altova und dessen Lieferanten dar. Die AltovaXML 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 AltovaXML Software bestehen, vorbehält. Das Eigentumsrecht von Altova erstreckt sich auch auf alle Bilder, Fotografien, Animationen, Videos, Audioaufzeichnungen, Musikstücke, Texte und „Applets“, die Teil der AltovaXML 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 AltovaXML 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 AltovaXML Software hergestellt wurden, gestattet. Es entstehen Ihnen daraus keinerlei Eigentumsrechte an dem betreffenden Warenzeichen. XMLSpy®, Authentic®, StyleVision®, MapForce®, UModel®, DatabaseSpy®, DiffDog®, SchemaAgent®, SemanticWorks®, MissionKit®, Markup Your Mind® und Altova® sind (in mehreren Ländern eingetragene) Warenzeichen von Altova GmbH. Unicode und das Logo von Unicode sind Warenzeichen von Unicode, Inc. Windows, Windows XP, Windows Vista und Windows 7 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 AltovaXML 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).

(c) Gemeinsame Einschränkungen.

(i) Sie sind nicht berechtigt, an der Software Reverse Engineering durchzuführen, die AltovaXML 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 AltovaXML 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 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 AltovaXML 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. Sie sind nicht berechtigt, die AltovaXML Software (zur Gänze

 © 2012 Altova GmbH

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 diesem Lizenzvertrag ausdrücklich gestattet.

(ii) Sofern dies nicht ausdrücklich in diesem Vertrag gestattet ist, dürfen Sie die AltovaXML Software nicht kopieren, vertreiben oder andere Software davon ableiten. Alle Kopien, zu deren Anfertigung Sie gemäß diesem Lizenzvertrag berechtigt sind, müssen die im Original enthaltenen urheber- und patentrechtlichen Angaben sowie alle anderen Angaben hinsichtlich geistiger Eigentumsrechte unverändert enthalten. Die AltovaXML Software darf nicht verändert, adaptiert oder übersetzt werden. Sie dürfen weder direkt noch indirekt ein Pfand- oder Sicherungsrecht an der AltovaXML Software bestellen oder die Bestellung eines solchen zulassen. Es ist Ihnen nicht gestattet, wissentlich eine Handlung vorzunehmen, durch die die AltovaXML Software öffentlich zugänglich wird, oder die AltovaXML 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 AltovaXML Software einzuhalten. Sie sind verpflichtet, Ihre Mitarbeiter und Vertreter, die Zugriff auf die AltovaXML Software haben, von den in diesem Lizenzvertrag enthaltenen Beschränkungen in Kenntnis zu setzen und diese auf sie zu überbinden. Es ist Ihnen nicht gestattet die AltovaXML Software zu verändern oder zu modifizieren oder einen neuen Installer für die AltovaXML Software zu erstellen Sie erklären sich damit einverstanden, Altova gegenüber allen Forderungen, Prozessen oder Klagen sowie den Anwaltsgebühren, die aufgrund der Verwendung oder des Vertriebs der Software und/oder Dokumentation durch Sie entstehen, schad- und klaglos zu halten.

(d) AltovaXML Software Aktivierung, Updates, Lizenzüberwachung und Datengebrauch.

(i) 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 AltovaXML Software Ihr internes Netzwerk verwenden. **Die AltovaXML 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 AltovaXML 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. Sie erklären sich damit einverstanden, dass die Verwendung von Freischaltcodes, welche nicht von Altova erstellt werden oder wurden und nicht rechtmäßig von Altova oder einem dazu berechtigten Wiederverkäufer erworben wurden, zum Zweck der Aktivierung oder Nutzung der AltovaXML Software die gewerblichen Schutzrechte von Altova sowie die Bestimmungen dieses Lizenzvertrages verletzt. Sie erklären sich weiters damit einverstanden, dass Versuche mit dem Zweck der Umgehung oder Deaktivierung der urheberrechtlichen Schutzmaßnahmen oder Lizenzmanagementsystemen von Altova die gewerblichen Schutzrechte von Altova sowie die Bestimmungen dieses Lizenzvertrages verletzen. Altova behält sich ausdrücklich das Recht vor, alle rechtlich verfügbaren sowie angemessenen Mittel zu ergreifen, um derartige Praktiken zu verhindern und entgangenen Gewinn, Schäden und Kosten zurückerstattet zu bekommen.**

(ii) 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.

(iii) Der gesamte Wortlaut der Datenschutzbestimmungen von Altova kann unter http://www.altova.com/de/privacy eingesehen werden. Die Datenschutzbestimmungen sind durch Bezugnahme Teil dieses Lizenzvertrags. Durch Ihre Zustimmung zu den Bestimmungen dieses Lizenzvertrags bzw. durch Benutzung der AltovaXML 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.

(iv) Hinweis für Benutzer in Europa. Bitte beachten Sie, dass die in Absatz 3(d) beschriebenen Informationen von Altova, Inc., einem Unternehmen mit Sitz in Beverly, Massachusetts, USA, oder seinen Tochterunternehmen, Zweigniederlassungen oder weltweit ansässigen autorisierten Partnern zum Zweck der Datenverarbeitung, Analyse und Überprüfung nach außerhalb der EU transferiert werden können. Sie werden darauf hingewiesen, dass in den USA ein Datenschutzmodell zur Anwendung kommt, das teils auf Gesetzen, teils auf Regierungsverordnungen und zum Teil auf Selbstregulierung beruht. Des Weiteren werden Sie darauf hingewiesen, dass der Rat der Europäischen Union befunden hat, dass dieses amerikanische Modell den in Artikel 25 der Datenrichtlinie der Europäischen Union (Richtlinie 95/46/EC, 1995 O.J. (L 281) 31) festgelegten Datenschutzbestimmungen nicht "in ausreichendem Ausmaß" Rechnung trägt. Laut Artikel 26 der Datenrichtlinie der Europäischen Union, dürfen persönliche Daten dann von der Europäischen Union in ein Drittland übertragen werden, wenn die jeweilige Person ihre Zustimmung zur Übertragung derartiger Informationen eindeutig gegeben hat, unabhängig davon, in welchem Ausmaß diese Daten in anderen Ländern geschützt sind. Durch Ihre Zustimmung zu dieser Software-Lizenzvereinbarung gestatten Sie die Übertragung aller derartiger Informationen an die USA sowie deren Verarbeitung gemäß dieser Software-Lizenzvereinbarung und Altovas Datenschutbestimmungen.

(e) Keine weiteren Garantien, Haftungsausschluss. ES WIRD KEINE GARANTIE ODER GEWÄHRLEISTUNG FÜR DIE FEHLERFREIHEIT DER ALTOVAXML SOFTWARE ABGEGEBEN ODER GEWÄHRT UND VON ALTOVA KEINERLEI HAFTUNG DIESBEZÜGLICH ÜBERNOMMEN. UNABHÄNGIG VON IRGENDWELCHEN SUPPORTLEISTUNGEN NACH IRGENDEINEM TECHNISCHEN STANDARD IST DIE ALTOVAXML SOFTWARE KEINESFALLS FÜR DIE VERWENDUNG IN ODER IM ZUSAMMENHANG MIT KERNKRAFTANLAGEN, FLUGGERÄTENAVIGATION, KOMMUNIKATIONSSYSTEMEN ODER LUFTVERKEHRSKONTROLLEINRICHTUNGEN, MEDIZINISCHEN GERÄTEN ODER LEBENSERHALTUNGSSYSTEMEN, MEDIZINISCHEN ODER DIE GESUNDHEITSVORSORGE BETREFFENDEN ANWENDUNGEN, ODER SONSTIGEN ANWENDUNGEN KONZIPIERT, BEI DENEN FEHLER DER ALTOVAXML SOFTWARE ODER FEHLER BEI DER DATENVERARBEITUNG ZU TODESFÄLLEN, PERSONENSCHÄDEN ODER SCHWEREN SACH- ODER UMWELTSCHÄDEN FÜHREN KÖNNTEN. SIE ERKLÄREN SICH DAMIT EINVERSTANDEN, DASS SIE DIE ALLEINIGE VERANTWORTUNG FÜR DIE EIGNUNG UND ZWECKMÄSSIGKEIT DER ALTOVAXML SOFTWARE UND ALLE MIT DER SOFTWARE ERSTELLTEN ODER VERARBEITETEN DATEN FÜR DEN BEABSICHTIGTEN VERWENDUNGSZWECK TRAGEN UND SIE WERDEN ALTOVA UND DEREN GESCHÄFTSFÜHRER SOWIE ANGESTELLTE HINSICHTLICH ALLER ANSPRÜCHE, FORDERUNGEN UND KLAGEN DRITTER, DIE AUF DIE EIGNUNG UND ZWECKMÄSSIGKEIT DER VON IHNEN VERWENDETEN ALTOVAXML SOFTWARE ODER DIE VON DER ALTOVAXML SOFTWARE ERSTELLTEN DATEN GEGRÜNDET SIND, SCHAD- UND KLAGLOS HALTEN.

(f) Eingeschränkte Rechte und Exportbeschränkungen. Die Entwicklung der AltovaXML 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.

 © 2012 Altova GmbH

§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 AltovaXML Software und/oder Dokumentation darf ausschließlich unter Einhaltung aller anwendbaren Exportbestimmungen der Vereinigten Staaten von Amerika sowie des Landes, in dem Sie die AltovaXML Software erhalten haben, verwendet, exportiert oder wiedereingeführt werden. Insbesondere darf die AltovaXML 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 AltovaXML 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.

(g) Beendigung. Dieser Lizenzvertrag wird folgendermaßen beendet, ohne dass damit andere Rechte oder Rechtsbehelfe von Altova ausgeschlossen werden: (i) Indem Sie Altova eine schriftliche Benachrichtigung von der Vertragsbeendigung übermitteln, oder (ii) durch Altova, in Ausübung dessen Rechtes, in Form einer schriftlichen Benachrichtigung, oder (iii) durch Altova in Form einer diesbezüglichen schriftlichen Benachrichtigung, wenn Sie gegen diesen Lizenzvertrag verstoßen. Dieser Lizenzvertrag endet automatisch mit dem Ablauf des AltovaXML Lizenz-Zeitraums. Bei Beendigung des Lizenzvertrags dürfen Sie die gesamte AltovaXML Software, die unter diesem Lizenzvertrag lizenziert wurde, 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 AltovaXML Software in Ihrem Besitz oder Einflussbereich verbleiben bzw. im Zusammenhang mit dem Vertrieb Ihres Software-Produkts verfügbar sind. Die in den Abschnitten 1(e)-(f), 2(c)-(d), 3(c)-(i) niedergelegten Bestimmungen, soweit anwendbar, bleiben auch nach Beendigung dieses Lizenzvertrags weiterhin aufrecht.

(h) Software Dritter. Die dieser Lizenz unterliegende AltovaXML 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/de/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.

(i) Allgemeine Bestimmungen. 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 Lizenzvertrages ist nur in 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.

(i) Wenn Sie sich in der Europäischen Union befinden und die AltovaXML 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 AltovaXML 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.

(ii) Wenn Sie sich in den Vereinigten Staaten befinden und die AltovaXML 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 AltovaXML Software stehen, in die alleinige Zuständigkeit der einzel- und bundesstaatlichen Gerichte im Bundesstaat Massachusetts fallen.

(iii) Wenn Sie sich nicht in der Europäischen Union oder den Vereinigten Staaten befinden und die AltovaXML 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 AltovaXML 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.

Letzte Aktualisierung: 2012-04-11

 © 2012 Altova GmbH

Index

.

.NET-Schnittstelle,
Beispielcode, 84
Features, 4
Objekt Altova.AltovaXML.XMLValidator, 86
Objekt Altova.AltovaXML.XQuery, 100
Objekt Altova.AltovaXML.XSLT1, 89
Objekt Altova.AltovaXML.XSLT2, 94
Objektmodell, 84
Verwendung, 82, 84

A

Altova Erweiterungen,
Diagrammfunktionen (siehe Diagrammfunktionen), 128
Altova Erweiterungsfunktionen,
allgemeine Funktionen, 128
Diagrammfunktionen (siehe Diagrammfunktionen), 128
Altova XSLT 1.0-Prozessor,
Einschränkungen und implementierungsspezifisches Verhalten, 112
Altova XSLT 2.0-Prozessor,
allgemeine Informationen, 115
Informationen, 114
Altova.AltovaXML.Applikationsobjekt, 84
Altova.AltovaXML.dll, 7, 82
AltovaXML,
Befehlszeilenfeatures, 4
Benutzerhandbuch, 3
Dokumentation, 3
Einführung, 3
Features der COM-Schnittstelle, 4
Funktionsumfang, 5
Hauptfunktionen, 4
Installation, 7
Paket, 4
Systemanforderungen, 7
Verwendung, 10
AltovaXML registrieren,
als COM-Serverobjekt, 7
AltovaXML.jar, 41, 57
und CLASSPATH, 7
AltovaXMLLib.dll, 7, 41, 57
Atomisierung von Nodes,
in XPath 2.0- und XQuery 1.0-Auswertung, 122

B

Befehlszeile,
Features, 4
für XQuery 1.0 Ausführungen, 20
für XSLT 1.0 Transformationen, 16
für XSLT 2.0 Transformationen, 18
Hilfe, 11
Validierung und Wohlgeformtheitsprüfung, 13
Versionsinformationen, 11
Zusammenfassung Verwendung, 11
Beispiel, 84
Beispiele, 22, 37, 41, 57
Bibliotheksmodule,
in XQuery-Dokument, 118

C

C# Beispielcode,
für .NET-Schnittstelle, 84
C++-Beispielcode,
für COM-Schnittstelle, 39
CLASSPATH,
und AltovaXML.jar, 7
Collations,
in XPath 2.0, 122
in XQuery-Dokument, 118
COM interface,
XQuery interface, 34
COM Server,
freigeben, 107
COM Serverobjekt,
AltovaXML registrieren, 11
com.altova-Prozessoren, 41, 57
COM-Schnittstelle,
Applikationsobjekt, 25
Beispielcode, 37
C++-Beispielcode, 39

© 2012 Altova GmbH

COM-Schnittstelle,
Features, 4
JScript-Beispielcode, 38
Objektmodell, 24
Validator-Schnittstelle, 26
Verwendung, 22
Visual Basic-Beispielcode, 37
XSLT1-Schnittstelle, 28
XSLT2-Schnittstelle, 31
COM-Serverobjekt,
AltovaXML registrieren, 7
Registrieren von AltovaXML, 23
Copyright, 136
Copyright-Informationen, 134
count() function in XPath 2.0,
siehe fn:count(), 122
count()-Funktion,
in XPath 1.0, 112

D

Datentypen,
in XPath 2.0 und XQuery 1.0, 122
deep-equal() function in XPath 2.0,
siehe fn:deep-equal(), 122
Default Functions Namespace,
für XPath 2.0- und XQueyr 1.0-Ausdrücke, 122
in XSLT 2.0 Stylesheets, 115
Dispatch-Schnittstelle,
Beschreibung, 22
Dokumentation,
Übersicht, 8
Dot NET,
siehe .NET, 82

E

Encoding,
in XQuery-Dokument, 118
Endbenutzer-Lizenzvereinbarung, 134
Erweiterungsfunktionen für XSLT und XQuery, 127
Evaluierungszeitraum, 135
von Altova Software-Produkten, 134
Externe Funktionen,
in XQuery-Dokument, 118

F

fn:base-uri in XPath 2.0,
Unterstützung in Altova-Prozessoren, 124
fn:collection in XPath 2.0,
Unterstützung in Altova-Prozessoren, 124
fn:count() in XPath 2.0,
und Whitespace, 122
fn:current-date in XPath 2.0,
Unterstützung in Altova-Prozessoren, 124
fn:current-dateTime in XPath 2.0,
Unterstützung in Altova-Prozessoren, 124
fn:current-time in XPath 2.0,
Unterstützung in Altova-Prozessoren, 124
fn:data in XPath 2.0,
Unterstützung in Altova-Prozessoren, 124
fn:deep-equal() in XPath 2.0,
und Whitespace, 122
fn:id in XPath 2.0,
Unterstützung in Altova-Prozessoren, 124
fn:idref in XPath 2.0,
Unterstützung in Altova-Prozessoren, 124
fn:index-of in XPath 2.0,
Unterstützung in Altova-Prozessoren, 124
fn:in-scope-prefixes in XPath 2.0,
Unterstützung in Altova-Prozessoren, 124
fn:last() in XPath 2.0,
und Whitespace, 122
fn:lower-case in XPath 2.0,
Unterstützung in Altova-Prozessoren, 124
fn:normalize-unicode in XPath 2.0,
Unterstützung in Altova-Prozessoren, 124
fn:position() in XPath 2.0,
und Whitespace, 122
fn:resolve-uri in XPath 2.0,
Unterstützung in Altova-Prozessoren, 124
fn:static-base-uri in XPath 2.0,
Unterstützung in Altova-Prozessoren, 124
fn:upper-case in XPath 2.0,
Unterstützung in Altova-Prozessoren, 124
Freigeben,
COM server, 107
functions,
XPath 2.0 and XQuery 1.0, 121

© 2012 Altova GmbH

Funktionen,
siehe unter XSLT 2.0-Funktionen, 117
von AltovaXML, 5

H

Hilfe,
über die Befehlszeile, 11

I

Implementierungsspezifisches Verhalten,
von XSLT 2.0.Funktionen, 117
Implizite Zeitzone,
und XPath 2.0-Funktionen, 122
Installation,
von AltovaXML, 7

J

Java-Klasse AltovaXMLFactory,
Beschreibung, 67
Java-Klasse XMLValidator,
Beschreibung, 69
Java-Klasse XQuery,
Beschreibung, 72
Java-Klasse XSLT1,
Beschreibung, 76
Java-Klasse XSLT2,
Beschreibung, 78
Java-Schnittstelle,
Beispielcode, 41, 57
Features, 4
Installation, 41, 57
Übersicht über die Klassen, 59
Übersicht über Klassen, 67
Verwendung, 41, 57
zusätzliche Dokumentation, 41, 57
Java-Schnittstelle IAltovaXMLEngine,
Beschreibung, 60
Java-Schnittstelle IAltovaXMLFactory,
Beschreibung, 61
Java-Schnittstelle IExecutable,
Beschreibung, 62
Java-Schnittstelle IReleasable,
Beschreibung, 62
Java-Schnittstelle IXMLValidator,
Beschreibung, 63
Java-Schnittstelle IXQuery,
Beschreibung, 64
Java-Schnittstelle IXSLT,
Beschreibung, 66
JScript-Beispielcode,
für COM-Schnittstelle, 38

L

last() function in XPath 2.0,
siehe fn:last(), 122
last()-Funktion,
in XPath 1.0, 112
Lizenz,
Informationen, 134
Löschen der Registrierung von AltovaXML als COM-Serverobjekt, 23

N

Namespaces,
im XSLT 2.0 Stylesheet, 115
in XQuery-Dokument, 118

O

OOXML-Dateien, 108

P

position() function,
in XPath 1.0, 112
position() function in XPath 2.0,
siehe fn:position(), 122
Prozessorinformationen, 110

© 2012 Altova GmbH

Q

QName Serialisierung,
Rückgabe durch XPath 2.0 -Funktionen, 124

R

Raw-Schnittstelle,
Beschreibung, 22
Rechte am geistigen Eigentum, 136
Rechtliches, 134
Registrieren von AltovaXML,
als COM-Serverobjekt, 23
Registrierung von AltovaXML,
als COM Serverobjekt, 11
Rückwärtskompatibilität,
des XSLT 2.0-Prozessors, 115

S

Schemafähigkeit,
des XPath 2.0- und XQuery-Prozessors, 122
Schema-Validierung eines XML-Dokuments,
für XQuery, 118
Scripts in XSLT/XQuery,
siehe Erweiterungsfunktionen, 127
Standardkonformität,
der Altova-Prozessoren, 110
des Altova XML Validators, 111

V

Validierung,
über die .NET-Schnittstelle, 86
über die Befehlszeile, 13
verfügbare Funktionen, 5
Versionsinformationen,
Befehlszeile, 11
Verwendung,
von AltovaXML, 10
Visual Basic-Beispielcode,
für COM-Schnittstelle, 37

W

Whitespace im XML-Dokument,
Behandlung durch den Altova XSLT 2.0-Prozessor, 115
Whitespace Nodes im XML-Dokument,
und Behandlung durch den XSLT 1.0-Prozessor, 112
Whitespace-Behandlung,
und XPath 2.0-Funktionen, 122
Wohlgeformtheitsprüfung,
über die .NET-Schnittstelle, 86
über die Befehlszeile, 13

X

XML-Validierung,
siehe Validierung, 86
XPath 2.0 Functions,
implementation information, 121
XPath 2.0-Funktionen,
allgemeine Informationen, 122
spezifische Funktionen siehe unter fn:, 122
XPath-Funktionen - Unterstützung,
für einzelne Funktionen siehe unter fn:, 124
XQuery,
Erweiterungsfunktionen, 127
XQuery 1.0 Functions,
implementation information, 121
XQuery 1.0 Transformationen,
über die .NET-Schnittstelle, 100
XQuery 1.0-Funktionen,
allgemeine Informationen, 122
spezifische Funktionen siehe unter fn:, 122
XQuery 1.0-Prozessor,
Informationen, 118
XQuery-Ausführung,
verfügbare Funktionen, 5
XQuery-Ausführungen,
über die BefehlszeileCommand line, 20
xs:QName,
siehe auch QName, 124
xsl:preserve-space, 112

© 2012 Altova GmbH

xsl:strip-space, 112
XSLT,
Erweiterungsfunktionen, 127
XSLT 1.0 Transformationen,
über die .NET-Schnittstelle, 89
über die Befehlszeile, 16
XSLT 1.0-Prozessor,
Einschränkungen und implementierungsspezifisches Verhalten, 112
XSLT 2.0 Stylesheet,
Namespace-Deklarationen, 115
XSLT 2.0 Transformationen,
über die .NET-Schnittstelle, 94
über die Befehlszeile, 18
XSLT 2.0-Funktionen,
implementierungsspezifisches Verhalten, 117
spezifische Funktionen siehe unter fn:, 117
XSLT 2.0-Prozessor,
allgemeine Informationen, 115
Informationen, 114
XSLT-Transformationen,
verfügbare Funktionen, 5

Z

Zeichen-Entities,
in der HTML-Ausgabe von XSLT-Transformationen, 112
Zeichennormalisierung,
in XQuery-Dokument, 118
ZIP-Dateien, 108

© 2012 Altova GmbH