AltovaXML 2007

Benutzer- und Referenzhandbuch

AltovaXML 2007 Benutzer- und Referenzhandbuch

Published: 2007

Inhaltsverzeichnis

Kapitel 1

Einführung

1 Einführung

AltovaXML 2007 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.

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.

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-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.

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.

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.

1.3 Systemanforderungen und Installation

Systemanforderungen
AltovaXML wird auf Windows NT, Windows 2000, Windows XP und Windows Server 2003 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.

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 (`AltovaXML2007.exe`) auf Ihren Rechner heruntergeladen haben, doppelklicken Sie darauf, um die Installation zu starten. AltovaXML wird unter `Programme` im Ordner `Altova/AltovaXML2007` 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 der Pfad zur Datei `AltovaXML.jar` (installiert im Ordner `Altova/AltovaXML2007`) muss als COM-Serverobjekt zur `CLASSPATH`-Registrierung hinzugefügt werden (dies erfolgt automatisch während der Installation). Bei der Installation wird auch die Datei `AltovaXMLLib.dll` im Verzeichnis `WINDIR\system32\` gespeichert. Beachten Sie jedoch: Wenn Sie nach der Installation den Pfad zur Datei `AltovaXML_COM.exe` ändern, müssen Sie sie 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.

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.

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"`.

2.1.1 XML-Validierung und Wohlgeformtheit

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

```
AltovaXML -validate XML-Datei [-schema Schemadatei | -dtd DTD-Datei
   ]
```

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.

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.

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.

Beispiele

- `AltovaXML -validate test.xml -schema testschema.xml`
- `AltovaXML -v test.xml -dtd testdtd.dtd`
- `AltovaXML -wellformed test.xml`
- `AltovaXML -w test.xml`

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.

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 -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.

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 -xslt2 test.xslt -in test.xml -out testout.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`.

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.

2.2.1 Registrieren von AltovaXML als COM Serverobjekt

Wenn Sie AltovaXML 2007 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.

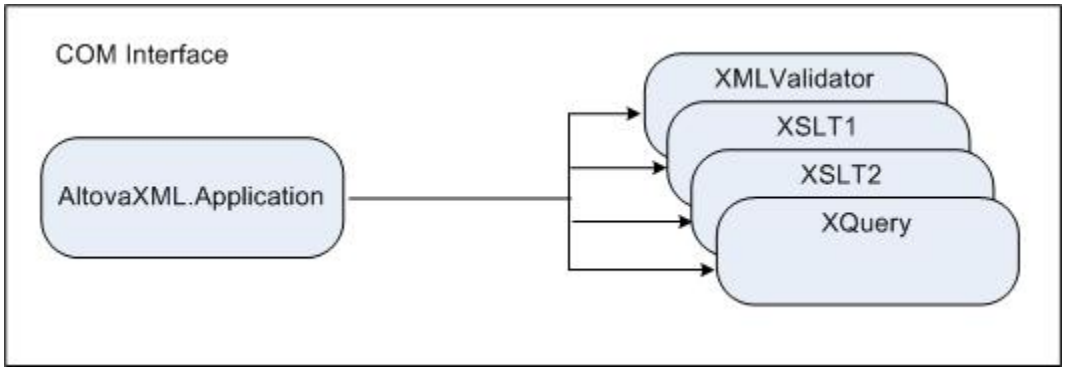
Registrierung manuell löschen
Wenn Sie AltovaXML_COM.exe manuell registriert haben und diese Registrierung nun löschen möchten, müssen Sie dies manuell tun. Um die Registrierung von AltovaXML manuell zu löschen, rufen Sie die Applikation über den Parameter /unregserver auf. Wenn sich die AltovaXML exe-Datei 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.

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

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.

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

`IsValid`
`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`
`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.

`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.

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`
`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.

`ExecuteAndGetResultAsString`
`ExecuteAndGetResultAsString` führt eine XSLT 1.0 Transformation aus und gibt das Ergebnis als Textstring zurück.

`AddExternalParameter`
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. Bei Angabe eines externen Parameters mit dem Namen eines existierenden (nicht gelöschten) Parameters wird ein Fehler ausgegeben. Da Parameterwerte XPath-Ausdrücke sind, müssen Parameterwerte, die Strings sind, in einfache Anführungszeichen eingeschlossen werden. *Siehe Beispiel 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.

`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.

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
```

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.

Methoden
Es stehen die folgenden Methoden zur Verfügung:

`Execute`
`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.

`ExecuteAndGetResultAsString`
`ExecuteAndGetResultAsString` führt eine XSLT 2.0 Transformation aus und gibt das Ergebnis als Textstring zurück.

`AddExternalParameter`
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. Bei Angabe eines externen Parameters mit dem Namen eines existierenden (nicht gelöschten) Parameters wird ein Fehler ausgegeben. 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()`).

`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.

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'
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`
`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.

`ExecuteAndGetResultAsString`
`ExecuteAndGetResultAsString` führt eine XQuery 1.0-Transformation aus und gibt das Ergebnis als Textstring zurück.

`AddExternalVariable`
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.

`AddExternalVariableAsXPath`
Nimmt einen Variablennamen und den Wert dieser Variable als Input-Argumente. Ist ähnlich wie die Methode `AddExternalVariable`, mit der Ausnahme, dass `AddExternalVariableAsXPath` zuerst 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.

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.

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
```

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;
}
```

2.3 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 `AltovaXML2007` Anwendungsordner finden, verfügbar.

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

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 im Verzeichnis `WINDIR\system32\` 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!");
        }
    }
}
```


2.3.1 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.

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.

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.

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 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.

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.

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`

Methoden
Es sind die folgenden Methoden definiert.

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

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
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.

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 **getXMLValidatorInstance**()
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.

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.

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 String 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.

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

```
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.

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 String 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 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 **addExternalParameter**(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.

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.

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 String 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.

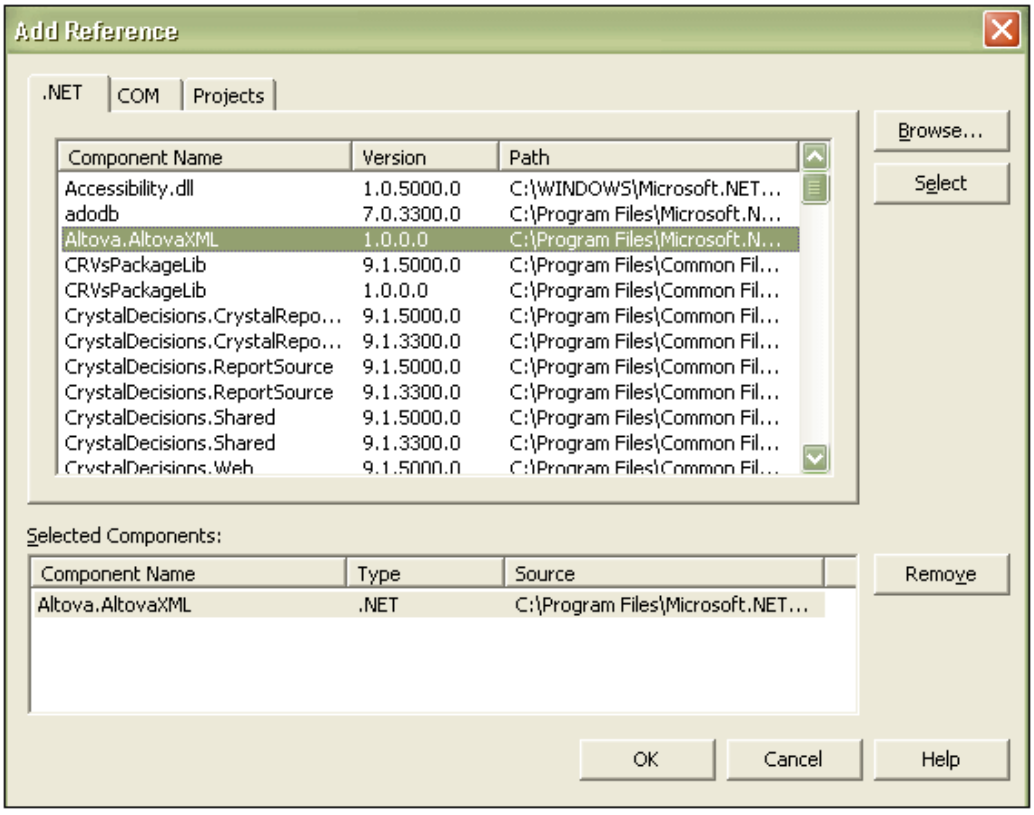
setXSLTStackSize
```
public void addExternalParameter(long nVal)
```
Die Stack Size (Stapelgröße) gibt die maximale Tiefe für die ausgeführten Anweisungen an. Wenn die Stapelgröße bei der Transformation überschritten wird, wird ein Fehler ausgegeben.
Definiert durch:
setXSLTStackSize in Schnittstelle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

2.3.3 .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.



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

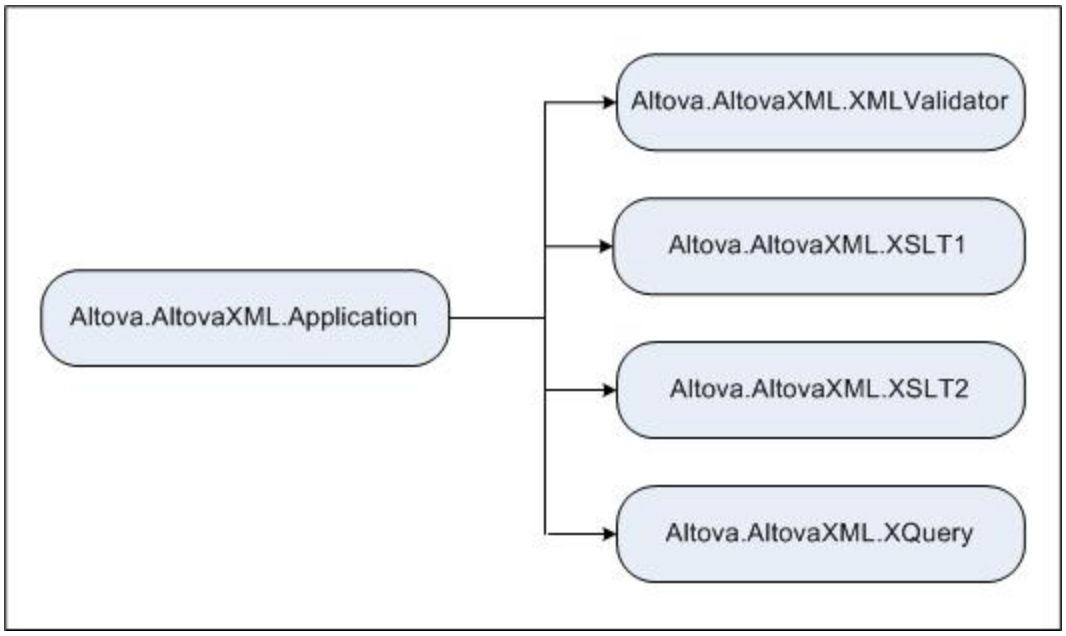
Registrieren von AltovaXML als COM-Serverobjekt
Die COM-Registierung erfolgt automatisch durch das AltovaXML Installationsprogramm. Wenn Sie den Ordner der Datei `AltovaXML_COM.exe` nach der Installation ändern, sollten Sie AltovaXML als COM-Serverobjekt registrieren. Führen Sie dazu den Befehl `AltovaXML`

_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.

Allgemeine Verwendung und Beispiel

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.



Beispiel

Die Verwendung der AltovaXML Klassen und Methoden im .NET Framework wird unten anhand eines Beispiels im C#-Code für ein Button Event gezeigt:

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

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.

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`
`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`
`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`
`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.

`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.

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
`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.

ExecuteAndGetResultAsString
`ExecuteAndGetResultAsString` führt eine XSLT 1.0 Transformation aus und gibt das Ergebnis als Textstring zurück.

AddExternalParameter
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. Bei Angabe eines externen Parameters mit dem Namen eines existierenden (nicht gelöschten) Parameters wird ein Fehler ausgegeben. Da Parameterwerte XPath-Ausdrücke sind, müssen Parameterwerte, die Strings sind, in einfache Anführungszeichen eingeschlossen werden.

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.

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.

Methoden
Es stehen die folgenden Methoden zur Verfügung:

`Execute`
`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.

`ExecuteAndGetResultAsString`
`ExecuteAndGetResultAsString` führt eine XSLT 2.0 Transformation aus und gibt das Ergebnis als Textstring zurück.

`AddExternalParameter`
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. Bei Angabe eines externen Parameters mit dem Namen eines existierenden (nicht gelöschten) Parameters wird ein Fehler ausgegeben. Da Parameterwerte XPath-Ausdrücke sind, müssen Parameterwerte, die Strings sind, in einfache Anführungszeichen eingeschlossen werden.

`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.

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`
`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.

`ExecuteAndGetResultAsString`
`ExecuteAndGetResultAsString` führt eine XQuery 1.0-Transformation aus und gibt das Ergebnis als Textstring zurück.

`AddExternalVariable`
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.

`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

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.

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. `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.

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.

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 Child-Textnode, der nur Whitespaces enthält und der zwischen zwei Elementen innerhalb eines mixed-content-Elements vorkommt). Dies kann sich auf den Wert auswirken, der von den Funktionen `fn:position()`, `fn:last()` und `fn:count()` zurückgegeben wird.

In jeder Node-Auswahl, bei der auch Text Nodes ausgewählt werden, sind normalerweise auch Boundary-whitespace-only Text Nodes enthalten. Da jedoch beim XML Infoset, das von den Altova-Prozessoren verwendet wird, boundary-whitespace-only Text Nodes entfernt werden, sind diese Nodes im XML Infoset nicht vorhanden. Folglich unterscheidet sich die Größe der Auswahl und die Nummierung der Nodes von einer, in der diese Text Nodes enthalten sind. Aus diesem Grund können sich die Ergebnisse der Funktionen `fn:position()`, `fn:last()` und `fn:count()` von denen anderer Prozessoren unterscheiden.

Am häufigsten tritt die Situation, dass boundary-whitespace-only Text Nodes als gleichrangige Elemente anderer Elemente überprüft werden, dann auf, wenn `xsl:apply-templates` verwendet wird, um Templates anzuwenden. Wenn die Funktionen `fn:position()`, `fn:last()` und `fn:count()` in Patterns mit einer Namensüberprüfung verwendet werden (z.B. `para[3], kurz für para[position()=3]`), sind boundary-whitespace-only Nodes irrelevant, da nur die benannten Elemente (`para` im obigen Beispiel) ausgewählt werden. (Beachten Sie jedoch, dass boundary-whitespace-only Nodes in Patterns, in denen ein Platzhalterzeichen wie z.B. `*[10]` verwendet wird, sehr wohl relevant sind).

Hinweis: Wenn der boundary-whitespace-only Text Node in der Ausgabe benötigt wird, fügen Sie das erforderliche Whitespace-Zeichen in eines der benachbarten Child-Elemente ein. So erzeugt z.B. das XML-Fragment:

```
<para>This is <b>bold</b> <i>italic</>. </para>
```

bei Verarbeitung mit dem XSLT Template

```
<xsl:template match="para">
   <xsl:apply-templates/>
</xsl:template>
```


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.
```

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.

3.3.1 Allgemeine Informationen

Der Altova XSLT 2.0-Prozessor entspricht der XSLT 2.0 Recommendation vom 23. Jänner 2007. Beachten Sie bitte die folgenden allgemeinen Informationen über den Prozessor.

Rückwärtskompatibilität
Der Altova XSLT 2.0-Prozessor ist rückwärtskompatibel. Die Rückwärtskompatibilität des XSLT 2.0.-Prozessors kommt nur dann zum Einsatz, wenn Sie den XSLT 2.0-Prozessor von Altova XML zur Verarbeitung eines XSLT 1.0 Stylesheets verwenden. Beachten Sie, dass sich das Ergebnis des XSLT 1.0-Prozessors und des rückwärtskompatiblen XSLT 2.0.Prozessors unter Umständen unterscheiden kann.

Bei allen anderen Altova-Produkten wird Rückwärtskompatibilität nie benötigt, da automatisch der entsprechende Prozessor für die Transformation ausgewählt wird. Angenommen, Sie legen fest, dass ein bestimmtes XML-Dokument in XMLSpy mit einem XSLT 1.0 Stylesheet verarbeitet werden soll. Bei Aufruf des Transformationsbefehls wählt XMLSpy zur Transformation automatisch den XSLT 1.0-Prozessor aus.

Anmerkung:
Diese Auswahl basiert auf dem Wert im Attribut `version` des Stylesheet-Elements `stylesheet` oder `transform`.

Namespaces
In Ihrem XSLT 2.0 Stylesheet sollten die folgenden Namespaces deklariert sein, damit Sie die in XSLT 2.0 verfügbaren Typ-Konstruktoren und Funktionen verwenden können. Normalerweise werden die unten aufgelisteten Präfixe verwendet; bei Bedarf können Sie auch andere Präfixe verwenden.

Namespace Name	Präfix	Namespace URI
XML Schema-Typen	`xs:`	`http://www.w3.org/2001/XMLSchema`
XPath 2.0-Funktionen	`fn:`	`http://www.w3.org/2005/xpath-functions`

Normalerweise werden diese Namespaces im Element `xsl:stylesheet` oder `xsl:transform` deklariert, wie unten gezeigt:

```
<xsl:stylesheet version="2.0"
   xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
   xmlns:xs="http://www.w3.org/2001/XMLSchema"
   xmlns:fn="http://www.w3.org/2005/xpath-functions"
   ...
</xsl:stylesheet>
```

Beachten Sie bitte die folgenden Punkte:

- Der Altova XSLT 2.0-Prozessor verwendet als Default Functions Namespace den Namespace für XPath 2.0- und XQuery 1.0-Funktionen (siehe Tabelle oben). Sie können daher XPath 2.0- und XSLT 2.0-Funktionen in Ihrem Stylesheet ohne Präfix verwenden. Wenn Sie den Namespace für XPath 2.0-Funktionen in Ihrem Stylesheet mit einem Präfix deklarieren, können Sie zusätzlich dazu das in der Deklaration zugewiesene Präfix verwenden.
- Bei Verwendung von Typ-Konstrukturoen und Typen aus dem XML Schema-Namespace, muss bei Aufruf des Typ-Konstruktors (z.B. xs:date) das in der

jeweiligen Namespace-Deklaration verwendeten Präfix verwendet werden.
- In den Candidate Recommendations vom 23. Jänner 2007 wurden die Datentypen untypedAtomic und duration (dayTimeDuration und yearMonthDuration), die sich zuvor im XPath Datatypes Namespace befanden (normalerweise mit dem Präfix xdt:) in den XML-Schema-Namespace verschoben.
- Einige XPath 2.0-Funktionen haben denselben Namen wie XML Schema-Datentypen. So gibt es z.B. für die XPath-Funktionen fn:string und fn:boolean XML-Schema-Datentypen mit demselben lokalen Namen: xs:string und xs:boolean. Wenn Sie daher den XPath-Ausdruck string('Hello') verwenden, wird der Ausdruck als fn:string('Hello') ausgewertet und nicht als xs:string('Hello').

Schemafähigkeit
Der Altova XSLT 2.0-Prozessor ist schemafähig.

Whitespaces im XML-Dokument
Standardmäßig entfernt der Altova XSLT 2.0-Prozessor alle boundary whitespaces aus boundary-whitespace-only Nodes im XML-Quelldokument. Dies wirkt sich auf die Werte aus, die die Funktionen `fn:position()`, `fn:last()`, `fn:count()` und `fn:deep-equal()` zurückgeben. Nähere Informationen dazu finden Sie in den Abschnitten über XPath 2.0- und XQuery 1.0-Funktionen unter Nur-Whitespace Nodes im XML-Quelldokument.

Hinweis: Wenn in der Ausgabe ein boundary-whitespace-only-Text Node erforderlich ist, fügen Sie das gewünschte Whitespace-Zeichen in eines der beiden benachbarten Child-Elemente ein. So erzeugt z.B. das XML-Fragment:

```
<para>This is <b>bold</b> <i>italic</>.</para>
```

bei Verarbeitung mit dem XSLT Template

```
<xsl:template match="para">
   <xsl:apply-templates/>
</xsl:template>
```

folgendes Resultat:

```
This is bolditalic.
```

Um in der Ausgabe ein Leerzeichen zwischen `bold` und `italic` zu erhalten, fügen Sie entweder in das Element `<b>` oder `<i>` in der XML-Quelle ein Leerzeichen ein. z.B.:

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

Wenn nun ein solches XML-Fragment mit demselben XSLT Template verarbeitet wird, erhalten Sie folgendes Ergebnis:

```
This is bold italic.
```

XSLT 2.0-Elemente und -Funktionen
Einschränkungen und implementierungsspezifisches Verhalten von XSLT 2.0-Elementen und -Funktionen werden im Abschnitt XSLT 2.0-Elemente und -Funktionen aufgelistet.

XPath 2.0-Funktionen
Implementierungsspezifisches Verhalten von XPath 2.0-Funktionen wird im Abschnitt XPath 2.0- und XQuery 1.0-Funktionen aufgelistet.

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.

`function-available`
Die Funktion überprüft, ob die XSLT 2.0-Funktionen zur Verfügung stehen und nicht ob die XPath 2.0-Funktionen verfügbar sind.

`unparsed-text`
Das Attribut `href` akzeptiert (i) relative Pfade für Dateien im Basis-URI-Ordner und (ii) absolute Pfade mit oder ohne das `file://`-Protokoll.

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. Jänner 2007. Der Query-Standard stellt frei, wie viele Funktionen zu implementieren sind. Im Folgenden finden Sie eine Liste, wie der Altova XQuery 1.0-Prozessor diese Funktionen implementiert.

Schemafähigkeit
Der Altova XQuery 1.0-Prozessor ist **schemafähig.**

Codierung
Die UTF-8 und die UTF-16 Zeichen-Kodierungen werden unterstützt.

Namespaces
Die folgenden Namespace-URIs und die damit verknüpften Bindings sind vordefiniert.

Namespace-Name	Präfix	Namespace URI
XML Schema-Typen	`xs:`	`http://www.w3.org/2001/XMLSchema`
Schema-Instanz	`xsi:`	`http://www.w3.org/2001/XMLSchema-instance`
Built-in Funktionen	`fn:`	`http://www.w3.org/2005/xpath-functions`
Lokale Funktionen	`local:`	`http://www.w3.org/2005/xquery-local-functions`

Beachten Sie bitte die folgenden Punkte:

- Der Altova XQuery 1.0-Prozessor ist so konfiguriert, dass die oben aufgelisteten Präfixe an die entsprechenden Namespaces gebunden sind.
- Da der oben angeführte Namespace für Built-in Funktionen der Default Functions Namespace in XQuery ist, muss beim Aufruf von built-in-Funktionen das Präfix `fn:` nicht verwendet werden (`string("Hello")` ruft z.B. die Funktion `fn:string` auf). Das Präfix `fn:` kann jedoch verwendet werden, um eine built-in-Funktion aufzurufen, ohne die Namespace im Abfrage-Prolog deklarieren zu müssen (z.B.: `fn:string("Hello")`).
- Sie können den Default Functions Namespace durch Deklarierung des `default function namespace`-Ausdrucks im Abfrageprolog ändern.
- Bei Verwendung von Typen aus dem XML Schema- und dem XPath-Datentypen-Namespace kann das Präfix `xs:` verwendet werden, ohne dass Sie den Namespace explizit deklarieren müssen und dieses Präfix im Abfrageprolog daran binden müssen. (Beispiele: `xs:date` und `xs:yearMonthDuration`.) Wenn Sie ein anderes Präfix verwenden möchten, muss dieses im Abfrageprolog für die Namespaces explizit deklariert werden. (Beispiel `declare namespace alt = "http://www.w3.org/2001/XMLSchema"; alt:date("2004-10-04")`.)
- Beachten Sie, dass die Datentypen `untypedAtomic`, `dayTimeDuration` und `yearMonthDuration` mit den Candidate Recommendations vom 23. Jänner 2007 aus dem XPath Datentypen-Namespace in den XML-Schema Namespace verschoben

wurden, d.h. `xs:yearMonthDuration`.

Wenn Namespaces für Funktionen, Typ-Konstrukturen, Node Tests usw. falsch zugewiesen wurden, wird ein Fehler ausgegeben. Beachten Sie jedoch, dass einige Funktionen denselben Namen wie Schema-Datentypen haben, z.B. `fn:string` und `fn:boolean`. (Sowohl `xs:string` als auch `xs:boolean` ist definiert.) Das Namespace-Präfix legt fest, ob die Funktion oder der Typ-Konstruktor verwendet wird.

XML-Quelldokument und Validierung
XML-Dokumente, die bei der Ausführung eines XQuery-Dokuments mit dem Altova XQuery 1.0-Prozessor verwendet werden, müssen wohlgeformt sein. Sie müssen jedoch nicht gemäß einem XML-Schema gültig sein. Wenn die Datei nicht gültig ist, wird die ungültige Datei ohne Schemainformationen geladen. Wenn die XML-Datei mit einem externen Schema verknüpft ist und gemäß diesem Schema gültig ist, werden für die XML-Daten nachträglich Validierungsinformationen generiert und für die Überprüfung der Abfrage verwendet.

Statische und dynamische Typ-Überprüfung
In der statischen Analysephase werden Aspekte der Abfrage überprüft wie z.B. die Syntax, ob externe Referenzen (z.B. für Module) vorhanden sind, ob aufgerufene Funktionen und Variablen definiert sind, usw. In der statischen Analysephase wird keine Typ-Überprüfung durchgeführt. Wenn in dieser Phase ein Fehler gefunden wird, wird eine Meldung ausgegeben und die Ausführung wird gestoppt.

Die dynamische Typ-Überprüfung wird in Laufzeit durchgeführt, während die Abfrage ausgeführt wird. Wenn ein Typ mit den Anforderungen einer Operation nicht kompatibel ist, wird ein Fehler ausgegeben. So gibt z.B. der Ausdruck `xs:string("1") + 1` einen Fehler zurück, weil die Operation "Addition" nicht an einem Operanden vom Typ `xs:string` ausgeführt werden kann.

Bibliotheksmodule
Bibliotheksmodule dienen zum Speichern von Funktionen und Variablen, damit diese wiederverwendet werden können. Der Altova XQuery 1.0-Prozessor unterstützt Module, die in einer **einzigen externen XQuery-Datei** gespeichert sind. Eine solche Moduldatei muss im Prolog eine `module`-Deklaration enthalten, in der ein Target Namespace zugewiesen wird. Hier ein Beispielmodul:

```
module namespace libns="urn:module-library";
declare variable $libns:company := "Altova";
declare function libns: webaddress() { "http://www.altova.com" };
```

Alle im Modul deklarierten Funktionen und Variablen gehören zu dem mit dem Modul verknüpften Namespace. Das Modul wird durch Import in eine XQuery-Datei mittels der `import module`-Anweisung im Abfrageprolog verwendet. Die `import module`-Anweisung importiert nur Funktionen und Variablen, die direkt in der Bibliotheksmodul-Datei deklariert sind:

```
import module namespace modlib = "urn:module-library" at
   "modulefilename.xq";
if    ($modlib:company = "Altova")
then   modlib:webaddress()
else   error("No match found.")
```

Externe Funktionen
Externe Funktionen, d.h. diejenigen Funktionen, die das Schlüsselwort `external` verwenden, werden nicht unterstützt:

```
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.

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. Jänner 2007.

3.5.1 Allgemeine Informationen

Standardkonformität

- Der Altova XPath 2.0-Prozessor implementiert die XPath 2.0 Recommendation vom 23. Jänner 2007. Der Altova XQuery 1.0-Prozessor implementiert die XQuery 1.0 Recommendation vom 23. Jänner 2007. Die Unterstützung der XPath 2.0- und XQuery 1.0-Funktionen in diesen beiden Prozessoren entspricht den Vorgaben in den XQuery 1.0 and XPath 2.0 Functions and Operators Recommendation vom 23. Jänner 2007.
- Der Altova XPath 2.0-Prozessor entspricht den Vorgaben für XML 1.0 (Fourth Edition) und XML Namespaces (1.0).

Default Functions Namespace
Der Default Functions Namespace wurde dem Standard entsprechend eingestellt. Funktionen können daher ohne Präfix aufgerufen werden.

Boundary-whitespace-only-Nodes in XML-Quelldokumenten
Aus den XML-Daten (und somit dem XML-Infoset), die an den Altova XPath 2.0- und den Altova XQuery 1.0-Prozessor übergeben werden, werden boundary-whitespace-only Text Nodes entfernt. (Ein boundary-whitespace-only Text Node ist ein Child-Textnode, der nur Whitespaces enthält und der zwischen zwei Elementen innerhalb eines mixed-content-Elements vorkommt). Dies kann sich auf den Wert auswirken, der von den Funktionen `fn:position()`, `fn:last()`, `fn:count()` und `fn:deep-equal()` zurückgegeben wird.

In jeder Node-Auswahl, bei der auch Text Nodes ausgewählt werden, sind normalerweise auch Boundary-whitespace-only Text Nodes enthalten. Da jedoch beim XML Infoset, das von den Altova-Prozessoren verwendet wird, boundary-whitespace-only Text Nodes entfernt werden, sind diese Nodes im XML Infoset nicht vorhanden. Folglich unterscheidet sich die Größe der Auswahl und die Nummerierung der Nodes von einer, in der diese Text Nodes enthalten sind. Aus diesem Grund können sich die Ergebnisse der Funktionen `fn:position()`, `fn:last()`, `fn:count()` und `fn:deep-equal()` von denen anderer Prozessoren unterscheiden.

Am häufigsten tritt die Situation, dass boundary-whitespace-only Text Nodes als gleichrangige Elemente anderer Elemente überprüft werden, dann auf, wenn `xsl:apply-templates` verwendet wird, um Templates anzuwenden. Wenn die Funktionen `fn:position()`, `fn:last()` und `fn:count()` in Patterns mit einer Namensüberprüfung verwendet werden (z.B. `para[3]`, kurz für `para[position()=3]`), sind boundary-whitespace-only Nodes irrelevant, da nur die benannten Elemente (`para` im obigen Beispiel) ausgewählt werden. (Beachten Sie jedoch, dass boundary-whitespace-only Nodes in Patterns, in denen ein Platzhalterzeichen wie z.B. `*[10]` verwendet wird, sehr wohl relevant sind).

Numerische Notation
Wenn bei der Ausgabe ein `xs:double` in einen String konvertiert wird, wird die logarithmische Darstellung (zB. `1.0E12`) verwendet, wenn der absolute Wert kleiner als 0,000001 oder größer als 1.000.000 ist. Andernfalls wird der Wert als Dezimalwert oder Integer angegeben.

Präzision von `xs:decimal`
Die Präzision bezieht sich auf die Anzahl der Stellen in einer Zahl. Laut Spezifikation sind mindestens 18 Stellen erforderlich. Bei Divisionen, bei denen ein Ergebnis vom Typ `xs:decimal` erzeugt wird, beträgt die Präzision 19 Kommastellen ohne Runden.

Implizite Zeitzone

Beim Vergleich zweier `date`, `time`, oder `dateTime` Werte muss die Zeitzone der verglichenen Werte bekannt sein. Wenn die Zeitzone in einem solchen Wert nicht explizit angegeben ist, wird die implizite Zeitzone verwendet. Als implizite Zeitzone wird die der Systemuhr verwendet. Der Wert kann mit Hilfe der Funktion `fn:implicit-timezone()` überprüft werden.

Collations
Es wird nur die Unicode Codepoint Collation unterstützt. Es können keine anderen Collations verwendet werden. Der Vergleich von Strings, u.a. auch für die Funktionen `fn:max` und `fn:min` basiert auf dieser Collation.

Namespace-Achse
Die Namespace-Achse wird in XPath 2.0 nicht mehr verwendet, wird aber weiterhin unterstützt. Um Namespace-Informationen mit XPath 2.0-Mechanismen aufzurufen, verwenden Sie die Funktionen `fn:in-scope-prefixes()`, `fn:namespace-uri()` und `fn:namespace-uri-for-prefix()`.

Static Type Extensions
Die optionale Funktion zum Überprüfen von statischen Typen wird nicht unterstützt.

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`	• Die Funktion `collection()` ist das Mapping einer Form `( string, nodes)`, wird derzeit als `available collections` bezeichnet und wird von der externen Umgebung leer gelassen. Die Funktion gibt daher entweder (i) die leere Sequenz zurück (bei Aufruf ohne ein Argument oder mit einer leeren Sequenz) oder (ii) einen Fehler (bei Aufruf mit einem nicht leeren Argument).
`count`	• Siehe Anmerkungen zu Whitespaces im Abschnitt Allgemeine Informationen.
`current-date, current-dateTime, current-time`	• Das aktuelle Datum und die aktuelle Uhrzeit werden aus der Systemuhr übernommen. • Die Zeitzone wird von der impliziten Zeitzone übernommen, die durch den Auswertungskontext geliefert wird; die implizite Zeitzone stammt von der Systemuhr. • Die Zeitzone wird im Ergebnis immer angegeben.
`deep-equal`	• Siehe Anmerkung zu Whitespace im Abschnitt Allgemeine Informationen.

doc	• Es wird nur dann ein Fehler ausgegeben, wenn unter dem angegebenen Pfad keine XML-Datei vorhanden ist oder wenn die Datei nicht wohlgeformt ist. Die Datei wird validiert, wenn ein Schema vorhanden ist. Wenn die Datei nicht gültig ist, so wird die ungültige Datei ohne Schema-Informationen geladen.
id	• In einem wohlgeformten aber ungültigen Dokument, das zwei oder mehr Elemente mit demselben ID-Wert enthält, wird das erste Element in der Dokumentenreihenfolge zurückgegeben.
in-scope-prefixes	• Im XML-Dokument dürfen nur Default Namespaces entdeklariert werden. Doch selbst wenn ein Default Namespace an einem Element-Node entdeklariert wird, wird das Präfix für den Default Namespace, welches der Nulllängenstring ist, für diesen Node zurückgegeben.
last	• Siehe Anmerkung zu Whitespace im Abschnitt Allgemeine Informationen.
lower-case	• Es wird nur der ASCII-Zeichensatz unterstützt.
normalize-unicode	• Im XML-Dokument dürfen nur Default Namespaces entdeklariert werden. Doch selbst wenn ein Default Namespace an einem Element-Node entdeklariert wird, wird das Präfix für den Default Namespace, welches der Nulllängenstring ist, für diesen Node zurückgegeben.
position	• Siehe Anmerkung zu Whitespace im Abschnitt Allgemeine Informationen.
resolve-uri	• Wenn das zweite, optionale Argument weggelassen wird, wird die aufzulösende URI (das erste Argument) gegen die Basis-URI aus dem statischen Kontext aufgelöst. Dies ist die URI des XSLT Stylesheets oder die im Prolog des XQuery-Dokuments angegebene Basis-URI. • Die relative URI (das erste Argument) wird nach dem letzten "/" in der Pfadnotation der Basis-URI-Notation angehängt. • Wenn der Wert des ersten Arguments der Nulllängenstring ist, wird die Basis-URI aus dem statischen Kontext zurückgegeben und diese URI inkludiert den Dateinamen des Dokuments, von dem die Basis-URI des statischen Kontexts abgeleitet wird (z.B. die XSLT- oder XML-Datei).
static-base-uri	• Die Basis-URI aus dem statischen Kontext ist die Basis-URI des XSLT Stylesheets oder die im Prolog des XQuery-Dokuments angegebene Basis-URI. • Bei Verwendung des XPath Evaluators in der XMLSpy IDE ist die Basis-URI aus dem statischen Kontext die URI des aktiven XML-Dokuments.
upper-case	• Es wird nur der ASCII-Zeichensatz unterstützt.

Kapitel 4

Lizenzvereinbarung

4 Lizenzvereinbarung

THIS IS A LEGAL DOCUMENT -- RETAIN FOR YOUR RECORDS
ALTOVA DEVELOPER LICENSE AGREEMENT
FOR ALTOVAXML SOFTWARE

Licensor:

Altova GmbH
Rudolfsplatz 13a/9
A-1010 Wien
Austria

Important – Read Carefully. Notice to User

This Altova Developer License Agreement ("DLA") governs your right to use, bundle, integrate and distribute AltovaXML software (the "Software"). Additional information about the Software can be found on the Altova Web Site. This DLA is a legal document between you and Altova GmbH ("Altova"). It is important that you read this document before using the Altova-provided software and any accompanying documentation, including, without limitation, printed materials, 'online' files, or electronic documentation ("Documentation"). By installing the Software, or including the Software in your application, or distributing the Software, or otherwise using the Software, you agree to be bound by the terms of this DLA as well as the Altova Privacy Policy ("Privacy Policy") including, without limitation, the warranty disclaimers, limitation of liability, data use and termination provisions below. You agree that this agreement is enforceable like any written agreement negotiated and signed by you. If you do not agree, you are not licensed to use or distribute the Software, and you must destroy any downloaded copies of the Software in your possession or control. Please go to our Web site at http://www.altova.com/ALTOVAXMLdla to download and print a copy of this DLA for your files and http://www.altova.com/privacy to review the privacy policy.

1. SOFTWARE LICENSE

(a) License Grant. Upon your acceptance of this DLA, Altova grants you a non-exclusive, non-transferable limited worldwide license to: (i) develop software applications that include the Software and/or Documentation, (ii) reproduce the Software and/or Documentation, and (iii) distribute the Software in executable form and Documentation in the manner hereinafter provided to end users for the purpose of being used in conjunction with a software application developed by you.

(b) Internal Use. You may install the Software on a server within your network for the purpose of downloading and installing the Software (to an unlimited number of client computers on your internal network).

(c) External Use. You may distribute the Software and/or Documentation to any third party electronically or via download from the website or on physical media such as CD-ROMS or diskettes as part of or in conjunction with products that you have developed.

(d) Distribution Restrictions. In addition to the restrictions and obligations provided in other sections of this DLA, your license to distribute the Software and/or Documentation is further subject to all of the following restrictions: (i) the Software and/or Documentation shall only be licensed and not sold; (ii) you may not make the Software and/or Documentation available as a stand alone product and if distributed as part of a product bundle you may charge for the product bundle provided that you license such product bundle at the same or lower fee at which you license any reasonably equivalent product bundle which does not include the Software; (iii) you must use the Software and/or Documentation provided by Altova AS IS and may not impair, alter or remove Altova's copyright or license statements or any other files; and (iv) other Altova products cannot be distributed or used under this DLA.

(e) Title. This DLA gives you a limited license to reproduce and distribute the Software and/or

Documentation. Altova and its suppliers retain all right, title and interest, including all copyright and intellectual property rights, in and to, the Software and/or Documentation and all copies thereof. All rights not specifically granted in this DLA are reserved by Altova.

(f) Reverse Engineering. You may not reverse engineer, decompile, disassemble or otherwise attempt to discover the source code, underlying ideas, underlying user interface techniques or algorithms of the Software by any means whatsoever, directly or indirectly, or disclose any of the foregoing, except to the extent you may be expressly permitted to decompile under applicable law if , it is essential to do so in order to achieve operability of the Software with another software program, and you have first requested Altova to provide the information necessary to achieve such operability and Altova has not made such information available. Altova has the right to impose reasonable conditions and to request a reasonable fee before providing such information. Any information supplied by Altova or obtained by you, as permitted hereunder, may only be used by you for the purpose described herein and may not be disclosed to any third party or used to create any software which is substantially similar to the expression of the Software.

(g) Additional Restrictions. You may not loan, rent, lease, sublicense, distribute or otherwise transfer all or any portion of the Software and/or Documentation to third parties except to the limited extent expressly provided herein. You may not copy, distribute or make derivative works of the Software and/or Documentation except as expressly set forth above, and any copies that you are permitted to make pursuant to this DLA must contain the same copyright, patent and other intellectual property markings that appear on or in the Software and/or Documentation. You may not alter, modify, adapt or translate the Software and/or Documentation or any part thereof. You may not, directly or indirectly, encumber or suffer to exist any lien or security interest on the Software; knowingly take any action that would cause the Software and/or Documentation to be placed in the public domain; or use the Software and/or Documentation in any computer environment not specified in this DLA. You will comply with applicable law and Altova's instructions regarding the use of the Software and/or Documentation. You agree to notify your employees and agents who may have access to the Software and/or Documentation of the restrictions contained in this DLA and to ensure their compliance with these restrictions. You agree to indemnify, hold harmless, and defend Altova from and against any claims or lawsuits, including attorney's fees that arise or result from your use or distribution of the Software and/or Documentation.

2. INTELLECTUAL PROPERTY RIGHTS

Acknowledgement of Altova's Rights. You acknowledge that the Software and/or Documentation and any copies that you are authorized by Altova to make are the intellectual property of and are owned by Altova and its suppliers. The structure, organization and code of the Software and/or Documentation are the valuable trade secrets and confidential information of Altova and its suppliers. The Software and/or Documentation is protected by copyright, including without limitation by United States Copyright Law, international treaty provisions and applicable laws in the country in which it is being used. You acknowledge that Altova retains the ownership of all patents, copyrights, trade secrets, trademarks and other intellectual property rights pertaining to the Software and/or Documentation, and that Altova's ownership rights extend to any images, photographs, animations, videos, audio, music, text and "applets" incorporated into the Software and/or Documentation and all accompanying printed materials. You will take no actions which adversely affect Altova's intellectual property rights in the Software and/or Documentation. Trademarks shall be used in accordance with accepted trademark practice, including identification of trademark owners' names. Trademarks may only be used to identify printed output produced by the Software, and such use of any trademark does not give you any right of ownership in that trademark., XMLSPY, AUTHENTIC, STYLEVISION, MAPFORCE, SCHEMAAGENT, DIFFDOG, UMODEL MARKUP YOUR MIND, AXAD, NANONULL, and ALTOVA are trademarks and/or registered trademark of Altova GmbH. Unicode and the Unicode Logo are trademarks of Unicode, Inc. Windows, Windows 95, Windows 98, Windows NT, Windows 2000 and Windows XP are trademarks of Microsoft. W3C, CSS, DOM, MathML, RDF, XHTML, XML and XSL are trademarks (registered in numerous countries) of the World Wide Web Consortium (W3C); marks of the

W3C are registered and held by its host institutions, MIT, INRIA and Keio. Except as expressly stated above, this DLA does not grant you any intellectual property rights in the Software and/or Documentation. Notifications of claimed copyright infringement should be sent to Altova's copyright agent as further provided on the Altova Web site.

3. WARRANTY DISCLAIMER AND LIMITATION OF LIABILITY

(a) THE SOFTWARE AND/OR DOCUMENTATION ARE PROVIDED TO YOU FREE OF CHARGE, AND ON AN "AS-IS" BASIS. ALTOVA PROVIDES NO TECHNICAL SUPPORT OR WARRANTIES FOR THE SOFTWARE AND/OR DOCUMENTATION. TO THE MAXIMUM EXTENT PERMITTED BY LAW, ALTOVA AND ITS SUPPLIERS DISCLAIM ALL WARRANTIES AND REPRESENTATIONS, WHETHER EXPRESS, IMPLIED OR OTHERWISE, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF FITNESS FOR A PARTICULAR PURPOSE; MERCHANTABILITY; SATISFACTORY QUALITY, INFORMATIONAL CONTENT, OR ACCURACY, QUIET ENJOYMENT, TITLE, AND NON- INFRINGEMENT. ALTOVA DOES NOT WARRANT THAT THE SOFTWARE IS ERROR-FREE OR WILL OPERATE WITHOUT INTERRUPTION. IF APPLICABLE LAW REQUIRES ANY WARRANTIES WITH RESPECT TO THE SOFTWARE, ALL SUCH WARRANTIES ARE LIMITED IN DURATION TO 30 DAYS FROM THE DATE OF INSTALLATION OR USE. SOME STATES DO NOT ALLOW THE EXCLUSION OF IMPLIED WARRANTIES, SO THE ABOVE EXCLUSION MAY NOT APPLY TO YOU. THIS WARRANTY GIVES YOU SPECIFIC LEGAL RIGHTS AND YOU MAY ALSO HAVE OTHER LEGAL RIGHTS THAT VARY FROM STATE TO STATE. YOU AGREE THAT YOU ARE SOLELY RESPONSIBLE FOR THE ACCURACY AND ADEQUACY OF THE SOFTWARE FOR YOUR INTENDED USE AND YOU WILL INDEMNIFY AND HOLD HARMLESS ALTOVA FROM ANY 3RD PARTY SUIT TO THE EXTENT BASED UPON THE ACCURACY AND ADEQUACY OF THE SOFTWARE AND/OR DOCUMENTATION IN YOUR USE. WITHOUT LIMITATION, THE SOFTWARE IS NOT INTENDED FOR USE IN HAZARDOUS ENVIRONMENTS REQUIRING FAIL-SAFE CONTROLS INCLUDING WITHOUT LIMITATION THE OPERATION OF NUCLEAR FACILITIES, AIRCRAFT NAVIGATION, COMMUNICATION SYSTEMS, AIR TRAFFIC CONTROL, LIFE SUPPORT, OR WEAPONS SYSTEMS, WHERE THE FAILURE OF THE SOFTWARE COULD LEAD TO DEATH, PERSONAL INJURY, OR SEVERE PHYSICAL OR ENVIRONMENTAL DAMAGE.

(b) TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, IN NO EVENT SHALL ALTOVA OR ITS SUPPLIERS BE LIABLE FOR ANY SPECIAL, INCIDENTAL, DIRECT, INDIRECT OR CONSEQUENTIAL DAMAGES WHATSOEVER (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF BUSINESS PROFITS, BUSINESS INTERRUPTION, LOSS OF BUSINESS INFORMATION, OR ANY OTHER PECUNIARY LOSS) ARISING OUT OF THE USE OF OR INABILITY TO USE THE SOFTWARE AND/OR DOCUMENTATION, OR ANY PROVISION OF THIS DLA, EVEN IF ALTOVA HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. WHERE LEGALLY, LIABILITY CANNOT BE EXCLUDED, BUT MAY BE LIMITED, ALTOVA'S LIABILITY AND THAT OF ITS SUPPLIERS SHALL BE LIMITED TO THE SUM OF FIVE DOLLARS (USD. $5.00) IN TOTAL. BECAUSE SOME STATES AND JURISDICTIONS DO NOT ALLOW THE EXCLUSION OR LIMITATION OF LIABILITY, THE ABOVE LIMITATION MAY NOT APPLY TO YOU. IN SUCH STATES AND JURISDICTIONS, ALTOVA'S LIABILITY AND THAT OF ITS SUPPLIERS SHALL BE LIMITED TO THE GREATEST EXTENT PERMITTED BY LAW. THE FOREGOING LIMITATIONS ON LIABILITY ARE INTENDED TO APPLY TO THE WARRANTIES AND DISCLAIMERS ABOVE AND ALL OTHER ASPECTS OF THIS DLA.

4. DATA USE

The terms and conditions of the Privacy Policy are set out in full at http://www.altova.com/privacy and are incorporated by reference into this DLA. By your acceptance of the terms of this DLA or use of the Software, you authorize the collection, use and

disclosure of information collected by Altova for the purposes provided for in this -DLA and/or the Privacy Policy as revised from time to time. European users understand and consent to the processing of personal information in the United States for the purposes described herein. Altova has the right in its sole discretion to amend this provision of the DLA and/or Privacy Policy at any time. You are encouraged to review the terms of the Privacy Policy as posted on the Altova Web site from time to time.

5. EXPORT RULES AND GOVERNMENT RESTRICTED RIGHTS

The Software was developed entirely at private expense and is commercial computer software provided with **RESTRICTED RIGHTS**. Use, duplication or disclosure by the U.S. Government or a U.S. Government contractor or subcontractor is subject to the restrictions set forth in this Agreement and as provided in FAR 12.211 and 12.212 (48 C.F.R. §12.211 and 12.212) or DFARS 227. 7202 (48 C.F.R. §227-7202) as applicable. Consistent with the above as applicable, Commercial Computer Software and Commercial Computer Documentation is licensed to U.S. government end users only as commercial items and only with those rights as are granted to all other end users under the terms and conditions set forth in this **DLA**. Manufacturer is Altova GmbH, Rudolfsplatz, 13a/9, A-1010 Vienna, Austria/EU. You may not use or otherwise export or re-export the Software or Documentation except as authorized by United States law and the laws of the jurisdiction in which the Software and/or Documentation was obtained. In particular, but without limitation, the Software and/or Documentation may not be exported or re-exported (i) into (or to a national or resident of) any U.S. embargoed country or (ii) to anyone on the U.S. Treasury Department's list of Specially Designated Nationals or the U.S. Department of Commerce's Table of Denial Orders. By using the Software and/or Documentation, you represent and warrant that you are not located in, under control of, or a national or resident of any such country or on any such list.

6. TERM AND TERMINATION

Without prejudice to any other rights or remedies of Altova, this DLA may be terminated (a) by you giving Altova written notice of termination; or (b) by Altova, for any or no reason, giving you written notice of termination or (c) Altova giving you written notice of termination if you fail to comply with the terms and conditions of the DLA. Upon any termination of this DLA, you must cease all use of the Software and/or Documentation, licensed hereunder, destroy all copies then in your possession or control and take such other actions as Altova may reasonably request to ensure that no copies of the Software and/or Documentation remain in your possession or control. The terms and conditions set forth in Sections 1 (e), (f), (g), 2,3, 5, 6 , and 7 survive termination of this agreement as applicable.

7. GENERAL PROVISIONS

If you are located in the European Union and are using the Software and/or Documentation in the European Union and not in the United States, then this DLA will be governed by and construed in accordance with the laws of the Republic of Austria (excluding its conflict of laws principles and the U.N. Convention on Contracts for the International Sale of Goods) and you expressly agree that exclusive jurisdiction for any claim or dispute with Altova or relating in any way to your use of the Software and/or Documentation resides in the Handelsgericht Wien (Commercial Court, Vienna) and you further agree and expressly consent to the exercise of personal jurisdiction in the Handelsgericht Wien (Commercial Court, Vienna) in connection with any such dispute or claim.

If you are located in the United States or are using the Software and/or Documentation in the United States then this DLA will be governed by and construed in accordance with the law of the Commonwealth of Massachusetts, USA (excluding its conflict of laws principles and the U.N. Convention on Contracts for the International Sale of Goods) and you expressly agree that exclusive jurisdiction for any claim or dispute with Altova or relating in any way to your use of the Software and/or Documentation resides in the federal or state courts of Massachusetts and you further agree and expressly consent to the exercise of personal jurisdiction in the federal or state courts of Massachusetts in connection with any such dispute or claim.

If you are located outside of the European Union or the United States and are not using the

Software and/or Documentation in the United States, then this DLA will be governed by and construed in accordance with the laws of the Republic of Austria (excluding its conflict of laws principles and the U.N. Convention on Contracts for the International Sale of Goods) and you expressly agree that exclusive jurisdiction for any claim or dispute with Altova or relating in any way to your use of the Software and/or Documentation resides in the Handelsgericht Wien (Commercial Court, Vienna) and you further agree and expressly consent to the exercise of personal jurisdiction in the Handelsgericht Wien (Commercial Court, Vienna) in connection with any such dispute or claim.
This DLA will not be governed by the conflict of law rules of any jurisdiction or the United Nations Convention on Contracts for the International Sale of Goods, the application of which is expressly excluded. This DLA contains the entire agreement and understanding of the parties with respect to the subject matter hereof, and supersedes all prior written and oral understandings of the parties with respect to the subject matter hereof. Any notice or other communication given under this DLA shall be in writing and shall have been properly given by either of us to the other if sent by certified or registered mail, return receipt requested, or by overnight courier to the address shown on Altova's Web site for Altova and the address shown in Altova's records for you, or such other address as the parties may designate by notice given in the manner set forth above. This DLA will bind and inure to the benefit of the parties and our respective heirs, personal and legal representatives, affiliates, successors and permitted assigns. The failure of either of us at any time to require performance of any provision hereof shall in no manner affect such party's right at a later time to enforce the same or any other term of this DLA. This DLA may be amended only by a document in writing signed by both of us. In the event of a breach or threatened breach of this DLA by either party, the other shall have all applicable equitable as well as legal remedies. Each party is duly authorized and empowered to enter into and perform this DLA. If, for any reason, any provision of this DLA is held invalid or otherwise unenforceable, such invalidity or unenforceability shall not affect the remainder of this DLA, and this DLA shall continue in full force and effect to the fullest extent allowed by law. The parties knowingly and expressly consent to the foregoing terms and conditions.

Last updated: 2005-06-28

Index

.

A

B

C

D

E

F

H

I

J

L

N

P

Q

R

S

V

W

X

Z